FAST NUCES, Islamabad

SOFTWARE ENGINEERING

TeachTrack - Faculty Evaluation & Feedback

Final Report

Group Members:
Maryum Fasih | i22-0756
Abeer Jawad | i22-1041
Areeba Riaz | i22-1244
Section: **BSCS-H**
Submitted to: **Ms. Rabail Zahid**

Contents

PROJECT INTRODUCTION

Introduction

TeachTrack by **Devinity** is a smart and intuitive **Faculty Evaluation & Feedback System** designed to enhance academic excellence through structured, anonymous, and data-driven feedback. It provides students with a seamless way to evaluate faculty performance, ensures faculty receive insightful reports for professional growth, and enables administrators to manage and analyze feedback efficiently. With a secure, user-friendly interface and powerful analytics, TeachTrack fosters transparency, accountability, and continuous improvement in teaching quality.

Project Vision

The goal of this project is to create an efficient, anonymous, and structured feedback system that allows students to evaluate faculty members based on predefined criteria. This system will assist in faculty performance assessment and improvement while ensuring the integrity and fairness of the feedback process.

Key Objectives:

- Provide an easy-to-use interface for students to submit feedback.
- Enable faculty members to view their feedback reports and improve teaching methods.
- Allow administrators to analyze faculty performance using reports and analytics.

Intended Use of the System

The system is designed for three key stakeholders: students, faculty members, and administrators.

- Students will use the system to submit anonymous feedback on faculty performance. They need a simple, intuitive interface that allows them to provide honest evaluations without fear of bias.
- Faculty Members will use the system to view feedback reports and identify areas for improvement in their teaching methods. They require structured, data-driven insights.
- Administrators will oversee the entire process, managing users, courses, and evaluation criteria to assess faculty performance and improve institutional quality.

FUNCTIONAL REQUIREMENTS

Authentication & Authorization

1. The system shall allow users (Admin, Faculty, and Students) to securely log in with a username and password.

2. The system shall verify credentials and redirect users to their respective dashboards based on their roles.

Student Functional Requirements

3. The system shall allow students to submit anonymous feedback through a structured form.

4. The system shall ensure student feedback cannot be traced back to individuals.

5. The system shall provide students with access to their profile settings and notification preferences.

6. The system shall allow students to view feedback-related analytics such as response statistics and course engagement summaries.

Faculty Functional Requirements

7. The system shall allow faculty to view individual and collective feedback on their courses.

8. The system shall allow faculty to filter feedback based on course, date, and type.

9. The system shall allow faculty to respond to feedback entries.

10. The system shall allow faculty to view graphical analytics such as average ratings and trends over time.

11. The system shall allow faculty to manage their profile settings and notification preferences.

Admin Functional Requirements

12. The system shall allow the admin to view a dashboard with total students, faculty, and courses.

13. The system shall allow the admin to manage users (add/edit/delete students and faculty).

14. The system shall allow the admin to manage courses (add/edit/delete course records).

15. The system shall allow the admin to view platform-wide feedback analytics and reports.

16. The system shall allow the admin to monitor system health and user statistics from the dashboard.

Analytics & Reporting

17. The system show analytics and summary to admin.

18. The system shall allow filtered analytics per role, course, or time frame.

System Management

19. The system shall ensure secure data handling, storage, and anonymization of feedback.

20. The system shall validate all inputs and provide meaningful error messages on failure.

21. The system shall send notifications (email or in-app) based on user preferences.

NON-FUNCTIONAL REQUIREMENTS

1. Product Requirements

- **Performance and Usability:** The system should efficiently support up to 500 users at the same time, with feedback submissions within three seconds. It must be capable of managing more than 10,000 records without compromising speed or stability.

- **Security:** All user passwords and sensitive data must be securely encrypted. Access to system features should be controlled based on user roles. Student feedback should remain anonymous/unanonymous based on user choice to maintain confidentiality, and administrative operations must be recorded for monitoring purposes.

- **Maintainability & Scalability:** The application should be built with a modular design to simplify future updates and improvements. It should also allow easy expansion to additional institutions with minimal configuration changes. Well-documented APIs must be available to support third-party integration.

2. Organizational Requirements

- **Compliance with Standards:** Development should adhere to the organization's established practices for coding, documentation, and version control.

- **Development Process:** The project should follow the SCRUM framework, using iterative development cycles and clearly defined user stories for progress tracking.

- **User Support:** Comprehensive user guides and training resources should be provided for all roles within the system, including students, faculty, and administrators.

3. External Requirements

- **Legal & Regulatory Compliance:** The system must comply with data protection laws such as GDPR (General Data Protection Regulation) to ensure user data is handled responsibly.
- **System Integration:** It should be compatible with widely used third-party services such as email platforms, and notification tools.
- **Platform Compatibility:** The application must work smoothly across common web browsers and operating systems.

USER STORIES

Admin User Stories

User Story 1: View Dashboard

- **Story:**
  "As an admin, I want to view the dashboard with an overview of key statistics (total students, faculty, courses) and quick access to different sections, so that I can monitor the system's overall health and access essential administrative tools easily."
- **Pre-Conditions:**
  - Admin is logged into the system.
  - Admin has the necessary permissions to access the dashboard.
- **Post-Conditions:**
  - The admin is presented with the dashboard containing the summary of key statistics and navigation options to other sections.
- **Sub-User Stories:**
  - **US1.1 – View Total Students:**
    "As an admin, I want to see the total number of students on the dashboard so that I can have a quick overview of the student count."
    ▪ **Pre-Conditions:** Admin is viewing the dashboard.
    ▪ **Post-Conditions:** Total student count is displayed on the dashboard.
  - **US1.2 – View Total Faculty:**
    "As an admin, I want to see the total number of faculty on the dashboard so that I can have a quick overview of the faculty count."

    - ▪ **Pre-Conditions:** Admin is viewing the dashboard.
    - ▪ **Post-Conditions:** Total faculty count is displayed on the dashboard.
  - **US1.3 – View Total Courses:**
    "As an admin, I want to see the total number of courses on the dashboard so that I can have a quick overview of the course offerings."
    - ▪ **Pre-Conditions:** Admin is viewing the dashboard.
    - ▪ **Post-Conditions:** Total course count is displayed on the dashboard.

User Story 2: Manage Courses

- **Story:**
  "As an admin, I want to manage courses by adding, editing, or deleting courses from the system, so that I can keep the courses list up-to-date and relevant for the students and faculty."
- **Pre-Conditions:**
  - Admin is logged in.
  - Admin has the necessary permissions to manage courses.
- **Post-Conditions:**
  - The courses list is updated as per the admin's actions (add, edit, delete).
- **Sub-User Stories:**
  - **US2.1 – Add New Course:**
    "As an admin, I want to add new courses to the system so that students and faculty can access the updated course offerings."
    - ▪ **Pre-Conditions:** Admin is on the manage courses page.
    - ▪ **Post-Conditions:** A new course is successfully added to the courses list.
  - **US2.2 – Edit Course Details:**
    "As an admin, I want to edit course details so that I can update course information as needed."
    - ▪ **Pre-Conditions:** Admin is on the manage courses page.
    - ▪ **Post-Conditions:** Course details are successfully updated.
  - **US2.3 – Delete Course:**
    "As an admin, I want to delete courses from the system so that outdated or irrelevant courses can be removed."
    - ▪ **Pre-Conditions:** Admin is on the manage courses page.
    - ▪ **Post-Conditions:** The selected course is successfully deleted from the courses list.

User Story 3: Manage Faculty

- **Story:**
  "As an admin, I want to manage faculty members by viewing, adding, editing, or removing faculty profiles, so that I can maintain an accurate list of faculties and ensure all faculty-related information is current."
- **Pre-Conditions:**
  - Admin is logged in.
  - Admin has the necessary permissions to manage faculty.
- **Post-Conditions:**
  - Faculty profiles are updated as per the admin's actions (add, edit, remove).
- **Sub-User Stories:**
  - **US3.1 – Add New Faculty:**
    "As an admin, I want to add new faculty members to the system so that their profiles are created and accessible."
    - **Pre-Conditions:** Admin is on the manage faculty page.
    - **Post-Conditions:** A new faculty profile is added successfully.
  - **US3.2 – Edit Faculty Details:**
    "As an admin, I want to edit faculty details so that their profiles are kept up to date."
    - **Pre-Conditions:** Admin is on the manage faculty page.
    - **Post-Conditions:** Faculty details are updated successfully.
  - **US3.3 – Remove Faculty:**
    "As an admin, I want to remove faculty members from the system so that the faculty list remains accurate and relevant."
    - **Pre-Conditions:** Admin is on the manage faculty page.
    - **Post-Conditions:** The selected faculty profile is removed successfully.

User Story 4: Manage Students

- **Story:**
  "As an admin, I want to manage student records by viewing, adding, editing, or removing students, so that I can ensure the system reflects up-to-date student information and can accommodate any changes as needed."
- **Pre-Conditions:**
  - Admin is logged in.
  - Admin has the necessary permissions to manage students.
- **Post-Conditions:**
  - Student records are updated as per the admin's actions (add, edit, remove).
- **Sub-User Stories:**

- **US4.1 – Add New Student:**
  "As an admin, I want to add new students to the system so that they are properly enrolled and listed in the student directory."
  - **Pre-Conditions:** Admin is on the manage students page.
  - **Post-Conditions:** A new student record is added successfully.
- **US4.2 – Edit Student Details:**
  "As an admin, I want to edit student details so that their profiles are kept up to date."
  - **Pre-Conditions:** Admin is on the manage students' page.
  - **Post-Conditions:** Student details are updated successfully.
- **US4.3 – Remove Student:**
  "As an admin, I want to remove students from the system so that their records are no longer included in the student directory."
  - **Pre-Conditions:** Admin is on the manage students' page.
  - **Post-Conditions:** The selected student record is removed successfully.

User Story 5: Update Profile

- **Story:**
  "As an admin, I want to view and update my personal profile, including my name, email, and password, so that I can ensure my account information is accurate and up to date for secure system access."
- **Pre-Conditions:**
  o Admin is logged in.
  o Admin has the necessary permissions to update their profile.
- **Post-Conditions:**
  o Admin's profile information is updated as per their inputs.
- **Sub-User Stories:**
  - **US5.1 – Change Admin Password:**
    "As an admin, I want to change my password to ensure my account remains secure."
    - **Pre-Conditions:** Admin is logged in.
    - **Post-Conditions:** Admin's password is successfully changed.
  - **US5.2 – Update Profile Information:**
    "As an admin, I want to update my name and email to ensure my profile is accurate."
    - **Pre-Conditions:** Admin is on the profile settings page.
    - **Post-Conditions:** Admin's profile information is updated successfully.

Student User Stories

User Story 1: Login Authentication

- **Story:**
  "As a student, I want to securely log in using my credentials so that I can access my personalized dashboard and services."
- **Pre-Conditions:**
  - Student has a registered account.
  - Login form is accessible on the platform.
- **Post-Conditions:**
  - Student is redirected to their dashboard upon successful login.
  - In case of failure, an appropriate error message is shown.

User Story 2: Submit Anonymous Feedback

- **Story:**
  "As a student, I want to submit feedback anonymously so that I can express my opinions without revealing my identity."
- **Pre-Conditions:**
  - Student is logged in.
  - Feedback form is available for the respective course or instructor.
- **Post-Conditions:**
  - Feedback is submitted without storing personal identifiers.
  - The system confirms submission.
- **Sub-User Stories:**
  - **US2.1 – Fill Feedback Form**:
    "As a student, I want to rate and comment using a structured form."
    ▪ Pre-Condition: Form is accessible.
    ▪ Post-Condition: Feedback is collected successfully.
  - **US2.2 – Confirm Anonymous Submission**:
    "As a student, I want to receive a confirmation after submitting my anonymous feedback."

▪ Pre-Condition: Feedback is submitted.
▪ Post-Condition: On-screen confirmation is shown.

User Story 3: Notification Preferences

- **Story:**
  "As a student, I want to manage notification settings so that I can receive updates relevant to my activities and interests."
- **Pre-Conditions:**
  - Student is logged in.
  - Notification system is configured.
- **Post-Conditions:**
  - Notification preferences are saved and reflected in future alerts.

User Story 4: Profile Management

- **Story:**
  "As a student, I want to view and update my profile details such as name, email, and password so that my account remains up to date and secure."
- **Pre-Conditions:**
  - Student is logged in.
  - Profile settings page is accessible.
- **Post-Conditions:**
  - Profile changes are saved and updated in the system.
- **Sub-User Stories:**
  - **US4.1 – Update Personal Info**
    "As a student, I want to edit my name or email when needed."
    ▪ Pre-Condition: Student is on the profile page.
    ▪ Post-Condition: Updated info is reflected.

User Story 5: View Analytics

- **Story:**
  "As a student, I want to view my learning analytics, such as feedback history and participation stats, so I can track my academic engagement."
- **Pre-Conditions:**
  - Student is logged in.

    - Analytics data is available.

- **Post-Conditions:**
    - Visual analytics are displayed.
    - Student can interact with graphs or tables to explore insights.

Faculty User Stories

User Story 1: View and Filter Feedback

- **Story:**
  "As a faculty member, I want to view and filter student feedback so that I can easily analyze relevant insights and address specific concerns."
- **Pre-Conditions:**
    - Faculty is logged in.
    - Feedback data is available for their course(s).
- **Post-Conditions:**
    - Filtered feedback list is displayed based on selected criteria (e.g., date, course, rating).
- **Sub-User Stories:**
    - **US1.1 – View All Feedback**:
      "As a faculty member, I want to view all feedback submitted for my courses."
    - **US1.2 – Filter by Course or Date**:
      "As a faculty member, I want to filter feedback to focus on specific sessions or timeframes."

User Story 2: Respond to Student Feedback

- **Story:**
  "As a faculty member, I want to respond to student feedback so that I can acknowledge concerns and improve communication."
- **Pre-Conditions:**
    - Faculty has access to feedback details.
    - Response functionality is enabled by the system.
- **Post-Conditions:**
    - Response is stored and optionally visible to relevant students or admins.

User Story 3: View Analytics

- **Story:**
  "As a faculty member, I want to check analytics based on the feedback I received so that I can track trends and evaluate my performance."
- **Pre-Conditions:**
  - Feedback data is collected and processed.
  - Faculty is logged in.
- **Post-Conditions:**
  - System displays visual analytics (charts, summaries) for selected feedback.

User Story 4: Manage Notification Settings

- **Story:**
  "As a faculty member, I want to manage my notification settings so that I receive alerts for new feedback or important system updates."
- **Pre-Conditions:**
  - Faculty is logged in.
  - Notification options are available in settings.
- **Post-Conditions:**
  - Notification preferences are updated.
  - Faculty receives relevant alerts accordingly.

User Story 5: Update Profile and Preferences

- **Story:**
  "As a faculty member, I want to update my profile and account settings so that my information is current and preferences are applied."
- **Pre-Conditions:**
  - Faculty is logged in.
  - Settings page is accessible.
- **Post-Conditions:**
  - Profile and settings are saved successfully.

PRODUCT BACKLOG

PRIORITIZATION CRITERIA

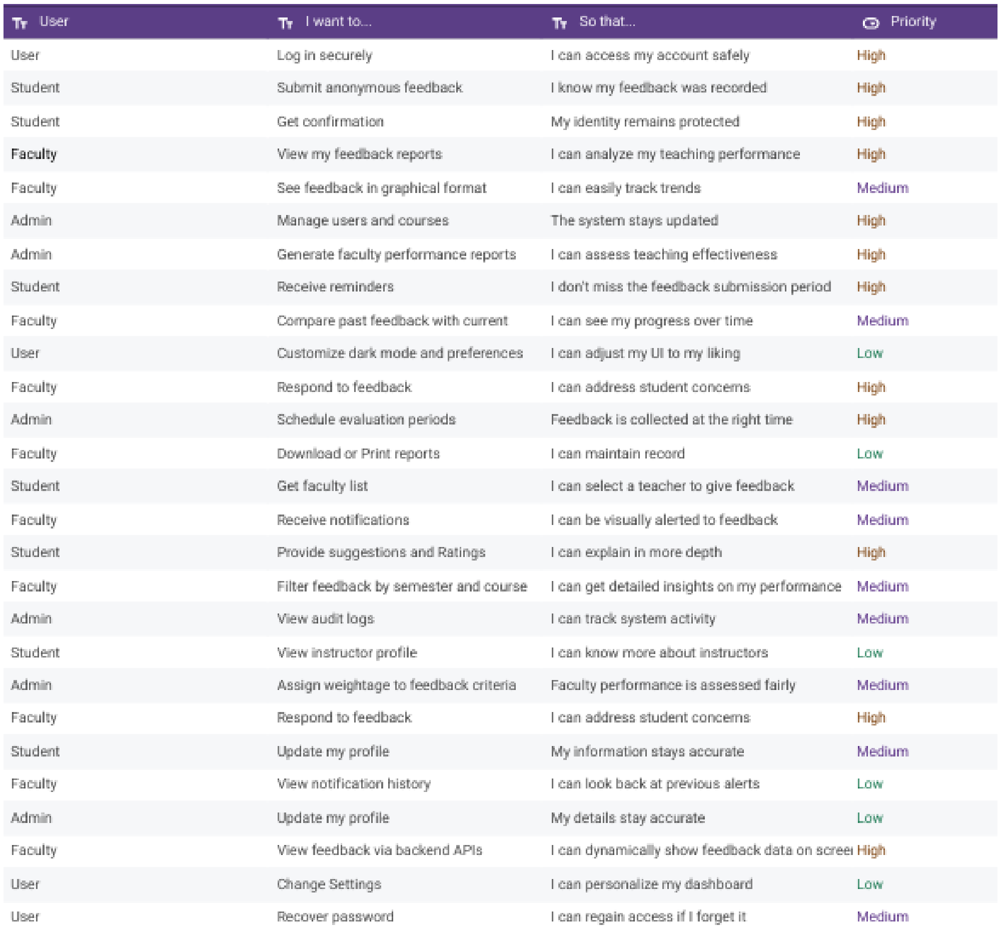
The product backlog is prioritized based on the following criteria:

1. **Must Have (High Priority):**
   Critical features necessary for system functionality, such as user authentication, feedback submission, and role-based access control. These are prioritized for early development.

2. **Should Have (Medium Priority):**
   Important features that enhance the system but are not essential for launch, like reporting tools, activity logs, and profile management.

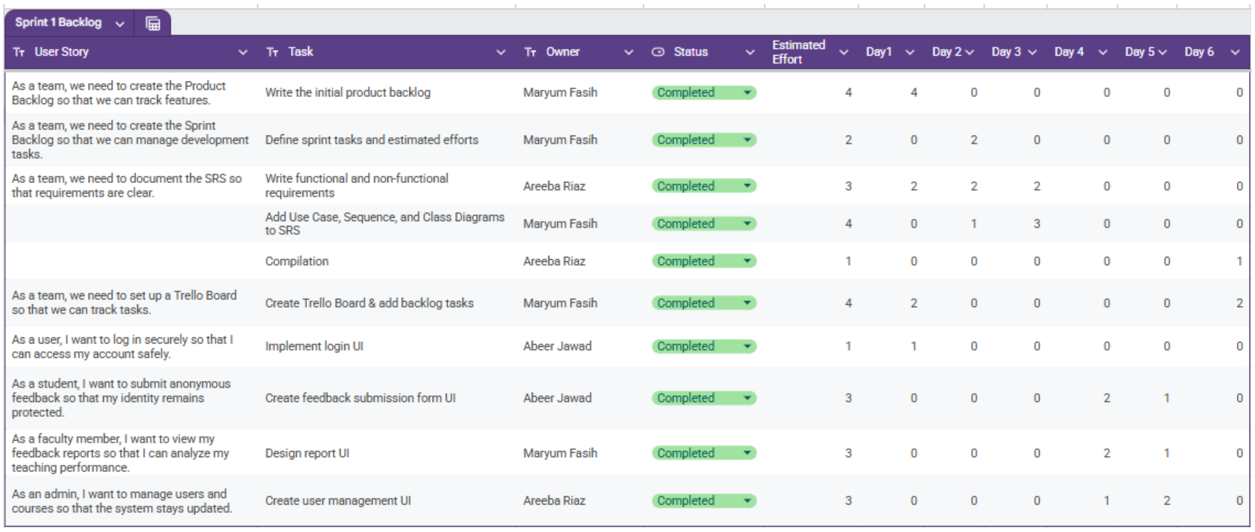
3. **Could Have (Low Priority):**
   non-essential features that improve user experience, such as dark mode toggling or report export options. These are developed after high and medium-priority items.

DETAILED BACKLOG ITEMS



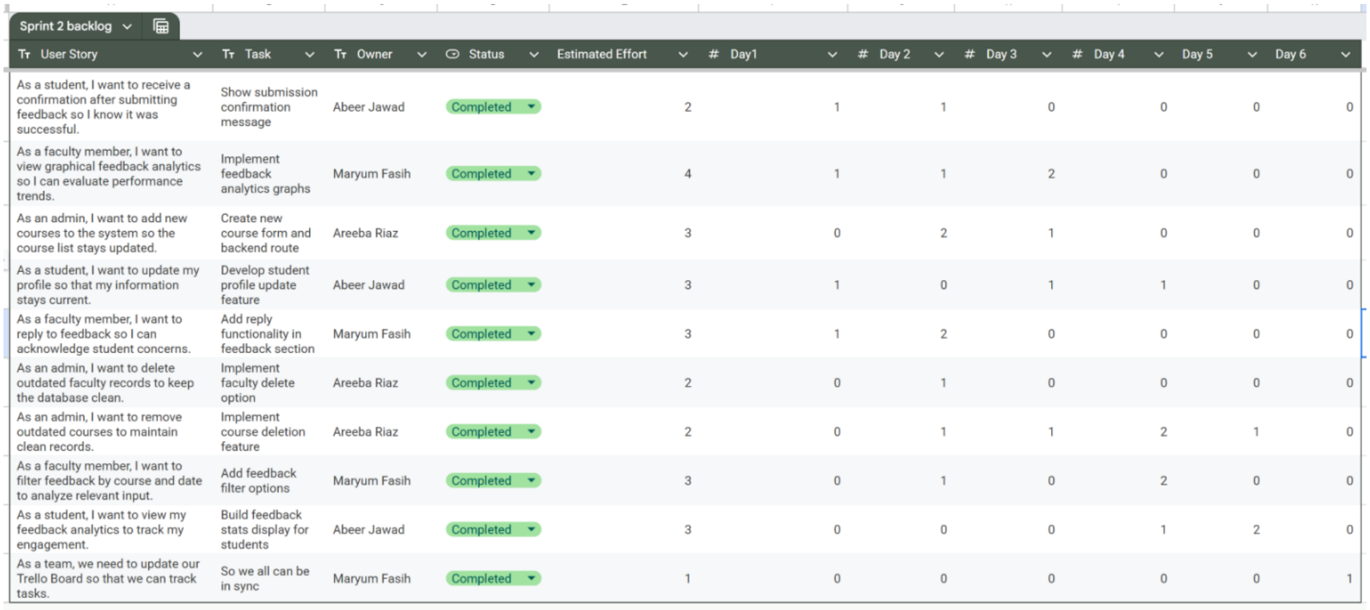
User	I want to...	So that...	Priority
User	Log in securely	I can access my account safely	High
Student	Submit anonymous feedback	I know my feedback was recorded	High
Student	Get confirmation	My identity remains protected	High
Faculty	View my feedback reports	I can analyze my teaching performance	High
Faculty	See feedback in graphical format	I can easily track trends	Medium
Admin	Manage users and courses	The system stays updated	High
Admin	Generate faculty performance reports	I can assess teaching effectiveness	High
Student	Receive reminders	I don't miss the feedback submission period	High
Faculty	Compare past feedback with current	I can see my progress over time	Medium
User	Customize dark mode and preferences	I can adjust my UI to my liking	Low
Faculty	Respond to feedback	I can address student concerns	High
Admin	Schedule evaluation periods	Feedback is collected at the right time	High
Faculty	Download or Print reports	I can maintain record	Low
Student	Get faculty list	I can select a teacher to give feedback	Medium
Faculty	Receive notifications	I can be visually alerted to feedback	Medium
Student	Provide suggestions and Ratings	I can explain in more depth	High
Faculty	Filter feedback by semester and course	I can get detailed insights on my performance	Medium
Admin	View audit logs	I can track system activity	Medium
Student	View instructor profile	I can know more about instructors	Low
Admin	Assign weightage to feedback criteria	Faculty performance is assessed fairly	Medium
Faculty	Respond to feedback	I can address student concerns	High
Student	Update my profile	My information stays accurate	Medium
Faculty	View notification history	I can look back at previous alerts	Low
Admin	Update my profile	My details stay accurate	Low
Faculty	View feedback via backend APIs	I can dynamically show feedback data on screei	High
User	Change Settings	I can personalize my dashboard	Low
User	Recover password	I can regain access if I forget it	Medium

SPRINT 1 BACKLOG



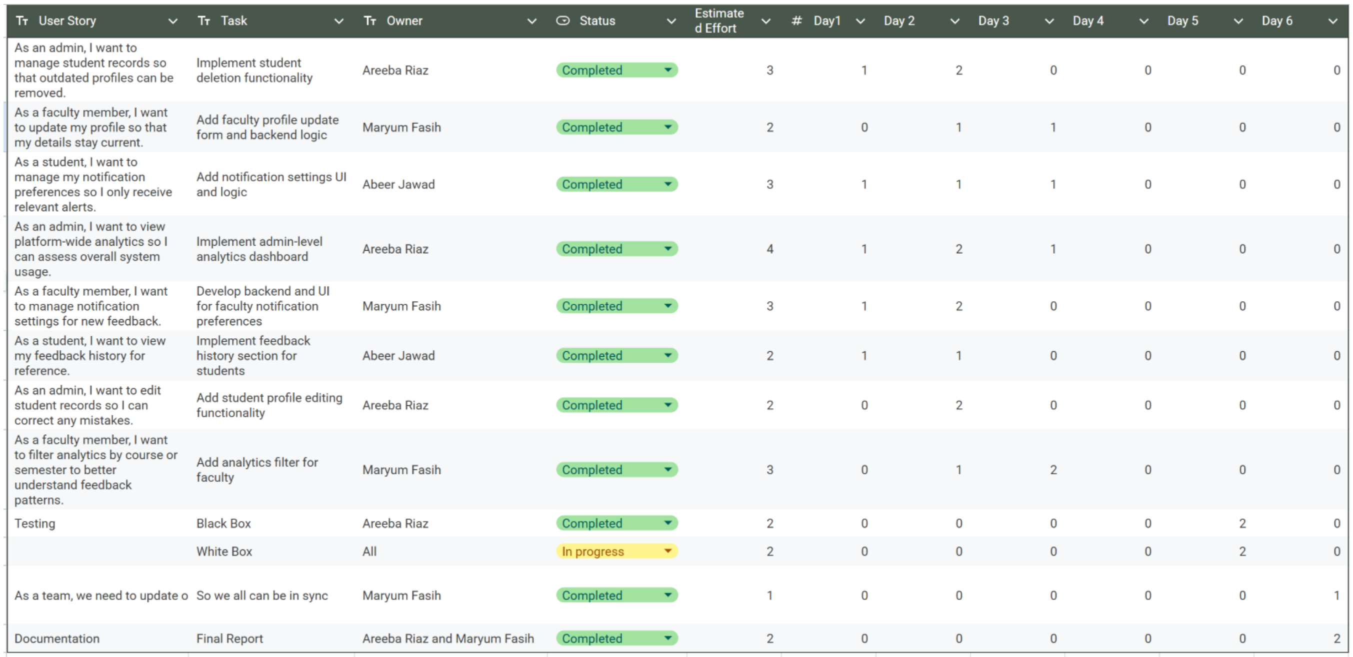
User Story	Task	Owner	Status	Estimated Effort	Day1	Day 2	Day 3	Day 4	Day 5	Day 6
As a team, we need to create the Product Backlog so that we can track features.	Write the initial product backlog	Maryum Fasih	Completed	4	4	0	0	0	0	0
As a team, we need to create the Sprint Backlog so that we can manage development tasks.	Define sprint tasks and estimated efforts	Maryum Fasih	Completed	2	0	2	0	0	0	0
As a team, we need to document the SRS so that requirements are clear.	Write functional and non-functional requirements	Areeba Riaz	Completed	3	2	2	2	0	0	0
	Add Use Case, Sequence, and Class Diagrams to SRS	Maryum Fasih	Completed	4	0	1	3	0	0	0
	Compilation	Areeba Riaz	Completed	1	0	0	0	0	0	1
As a team, we need to set up a Trello Board so that we can track tasks.	Create Trello Board & add backlog tasks	Maryum Fasih	Completed	4	2	0	0	0	0	2
As a user, I want to log in securely so that I can access my account safely.	Implement login UI	Abeer Jawad	Completed	1	1	0	0	0	0	0
As a student, I want to submit anonymous feedback so that my identity remains protected.	Create feedback submission form UI	Abeer Jawad	Completed	3	0	0	0	2	1	0
As a faculty member, I want to view my feedback reports so that I can analyze my teaching performance.	Design report UI	Maryum Fasih	Completed	3	0	0	0	2	1	0
As an admin, I want to manage users and courses so that the system stays updated.	Create user management UI	Areeba Riaz	Completed	3	0	0	0	1	2	0

SPRINT 2 BACKLOG



User Story	Task	Owner	Status	Estimated Effort	# Day1	# Day 2	# Day 3	# Day 4	Day 5	Day 6
As a student, I want to receive a confirmation after submitting feedback so I know it was successful.	Show submission confirmation message	Abeer Jawad	Completed	2	1	1	0	0	0	0
As a faculty member, I want to view graphical feedback analytics so I can evaluate performance trends.	Implement feedback analytics graphs	Maryum Fasih	Completed	4	1	1	2	0	0	0
As an admin, I want to add new courses to the system so the course list stays updated.	Create new course form and backend route	Areeba Riaz	Completed	3	0	2	1	0	0	0
As a student, I want to update my profile so that my information stays current.	Develop student profile update feature	Abeer Jawad	Completed	3	1	0	1	1	0	0
As a faculty member, I want to reply to feedback so I can acknowledge student concerns.	Add reply functionality in feedback section	Maryum Fasih	Completed	3	1	2	0	0	0	0
As an admin, I want to delete outdated faculty records to keep the database clean.	Implement faculty delete option	Areeba Riaz	Completed	2	0	1	0	0	0	0
As an admin, I want to remove outdated courses to maintain clean records.	Implement course deletion feature	Areeba Riaz	Completed	2	0	1	1	2	1	0
As a faculty member, I want to filter feedback by course and date to analyze relevant input.	Add feedback filter options	Maryum Fasih	Completed	3	0	1	0	2	0	0
As a student, I want to view my feedback analytics to track my engagement.	Build feedback stats display for students	Abeer Jawad	Completed	3	0	0	0	1	2	0
As a team, we need to update our Trello Board so that we can track tasks.	So we all can be in sync	Maryum Fasih	Completed	1	0	0	0	0	0	1

SPRINT 3 BACKLOG



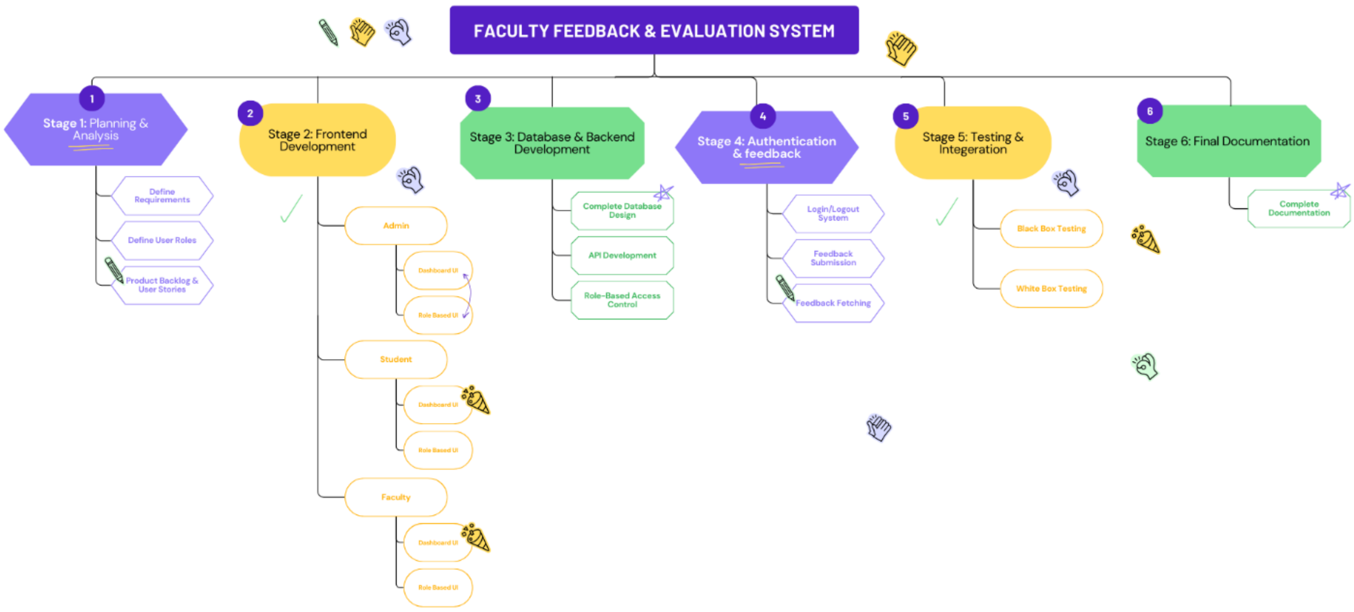
User Story	Task	Owner	Status	Estimated Effort	Day1	Day 2	Day 3	Day 4	Day 5	Day 6
As an admin, I want to manage student records so that outdated profiles can be removed.	Implement student deletion functionality	Areeba Riaz	Completed	3	1	2	0	0	0	0
As a faculty member, I want to update my profile so that my details stay current.	Add faculty profile update form and backend logic	Maryum Fasih	Completed	2	0	1	1	0	0	0
As a student, I want to manage my notification preferences so I only receive relevant alerts.	Add notification settings UI and logic	Abeer Jawad	Completed	3	1	1	1	0	0	0
As an admin, I want to view platform-wide analytics so I can assess overall system usage.	Implement admin-level analytics dashboard	Areeba Riaz	Completed	4	1	2	1	0	0	0
As a faculty member, I want to manage notification settings for new feedback.	Develop backend and UI for faculty notification preferences	Maryum Fasih	Completed	3	1	2	0	0	0	0
As a student, I want to view my feedback history for reference.	Implement feedback history section for students	Abeer Jawad	Completed	2	1	1	0	0	0	0
As an admin, I want to edit student records so I can correct any mistakes.	Add student profile editing functionality	Areeba Riaz	Completed	2	0	2	0	0	0	0
As a faculty member, I want to filter analytics by course or semester to better understand feedback patterns.	Add analytics filter for faculty	Maryum Fasih	Completed	3	0	1	2	0	0	0
Testing	Black Box	Areeba Riaz	Completed	2	0	0	0	0	2	0
	White Box	All	In progress	2	0	0	0	0	2	0
As a team, we need to update o	So we all can be in sync	Maryum Fasih	Completed	1	0	0	0	0	0	1
Documentation	Final Report	Areeba Riaz and Maryum Fasih	Completed	2	0	0	0	0	0	2

PROJECT PLAN

Work Breakdown Structure (WBS)

The Work Breakdown Structure (WBS) for the **Feedback Management System** divides the entire project into structured, manageable phases to streamline development, task assignment, and tracking. It begins with initial planning and expands into major development components such as frontend, backend, user-specific modules, authentication, feedback processing, reporting, testing, and deployment.

Each major area is broken down into actionable sub-tasks, ensuring clear roles, responsibility distribution, and timely completion. The WBS provides a hierarchical overview of all deliverables and dependencies, offering a practical roadmap for project coordination and execution.



Tech Stack:

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Java (Spring Boot, REST APIs)
- **Database:** MySQL

ARCHITECTURE DIAGRAM

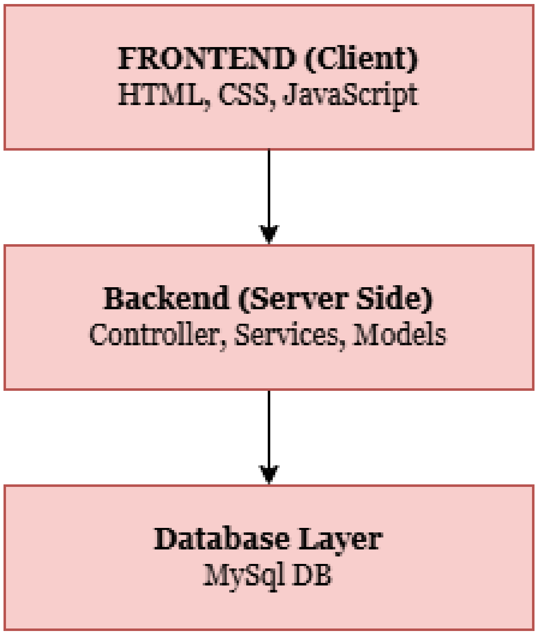
The system architecture is designed using a three-tier model comprising the **Frontend**, **Backend**, and **Database Layer**, ensuring modularity, maintainability, and scalability.

The **Frontend (Client)** is built using HTML, CSS, and JavaScript. It is responsible for presenting the user interface and handling user interactions. All user-facing features such as login, feedback forms, and dashboards are rendered and managed here.

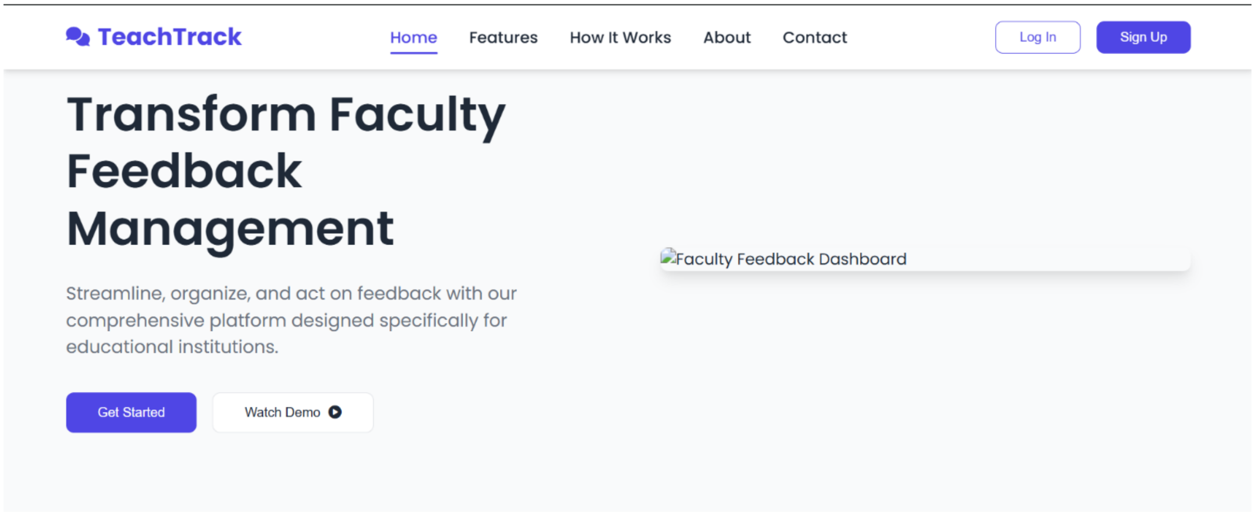
The **Backend (Server Side)** is developed using **Spring Boot (Java)**. It acts as the core engine of the application, managing the business logic through controllers, services, and models. It processes client requests, enforces role-based access control, and communicates with the database.

The **Database Layer** utilizes **MySQL**, which stores all persistent data including user details, feedback submissions, course records, and evaluation results. It supports relational queries and ensures data integrity across all system modules.

This layered architecture promotes a clear separation of concerns and enables easier testing, deployment, and future enhancements.

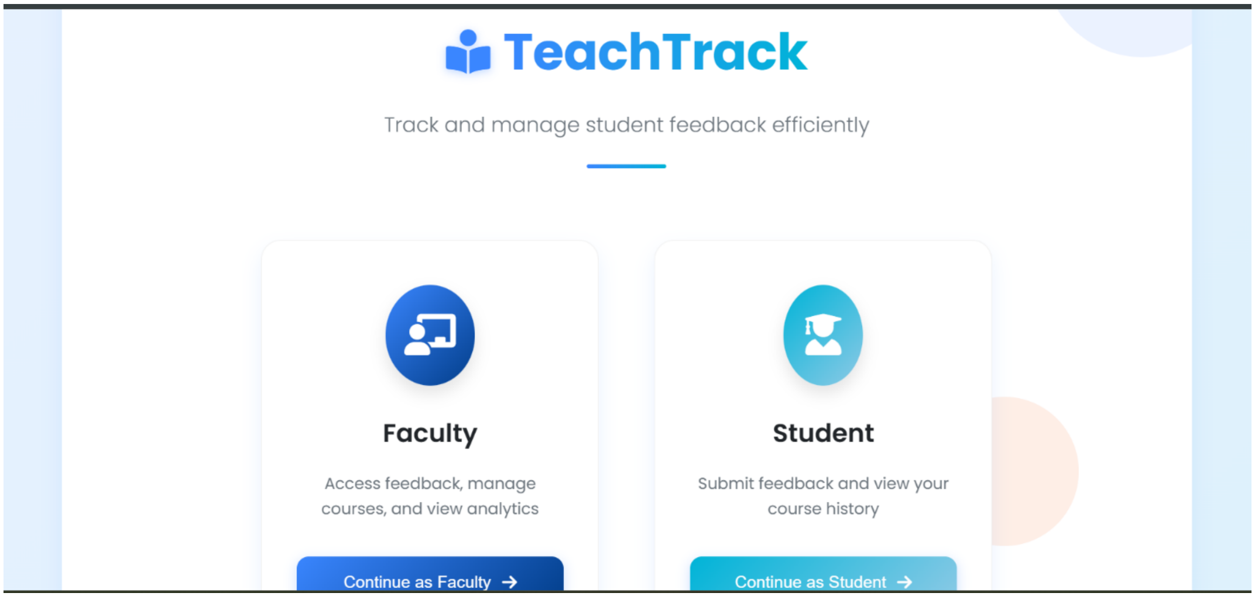


DESIGN (ALL SPRINT 3 ITEMS)



Transform Faculty Feedback Management

Streamline, organize, and act on feedback with our comprehensive platform designed specifically for educational institutions.



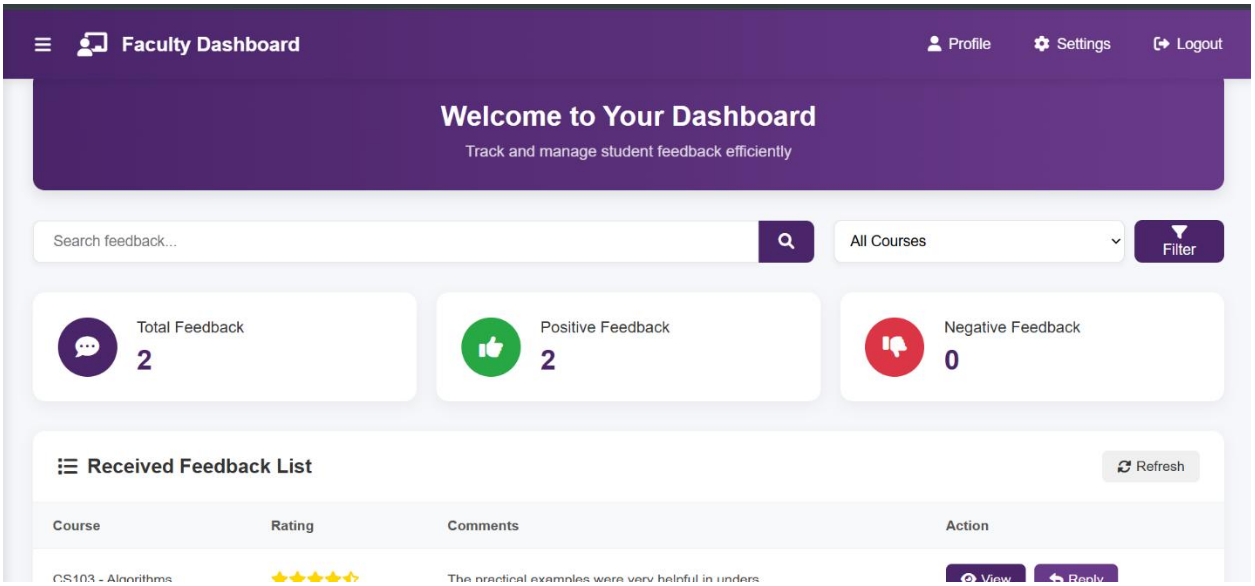
Track and manage student feedback efficiently

Faculty

Access feedback, manage courses, and view analytics

Student

Submit feedback and view your course history



Welcome to Your Dashboard

Track and manage student feedback efficiently

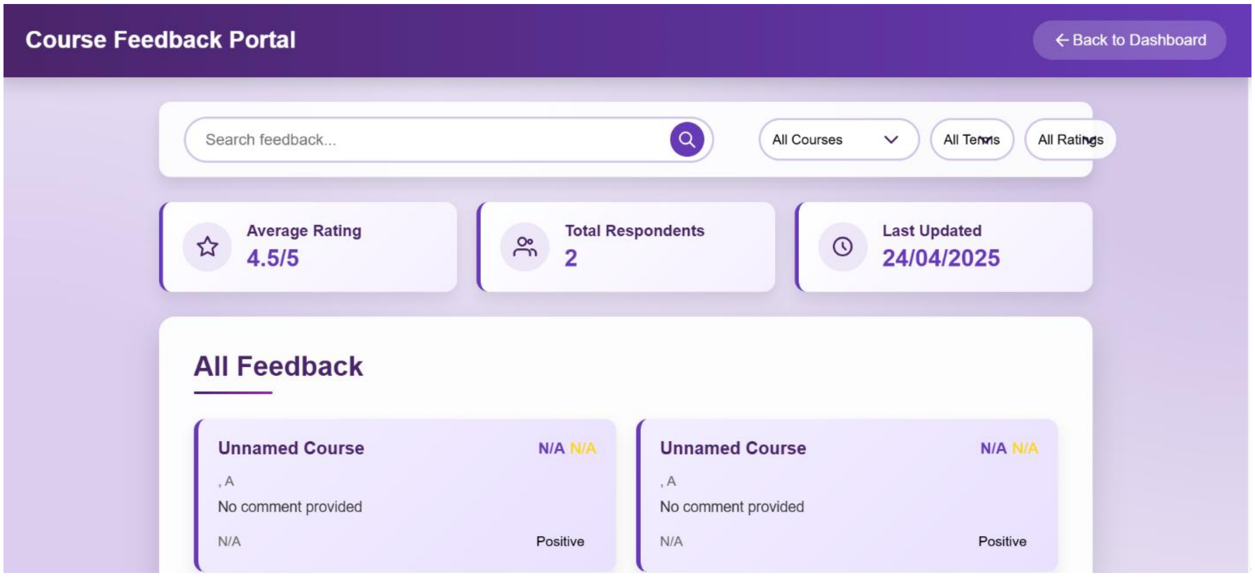
Total Feedback 2

Positive Feedback 2

Negative Feedback 0

Received Feedback List

Course	Rating	Comments	Action
CS103 - Algorithms	★★★★☆	The practical examples were very helpful in unders	View Reply



Average Rating 4.5/5

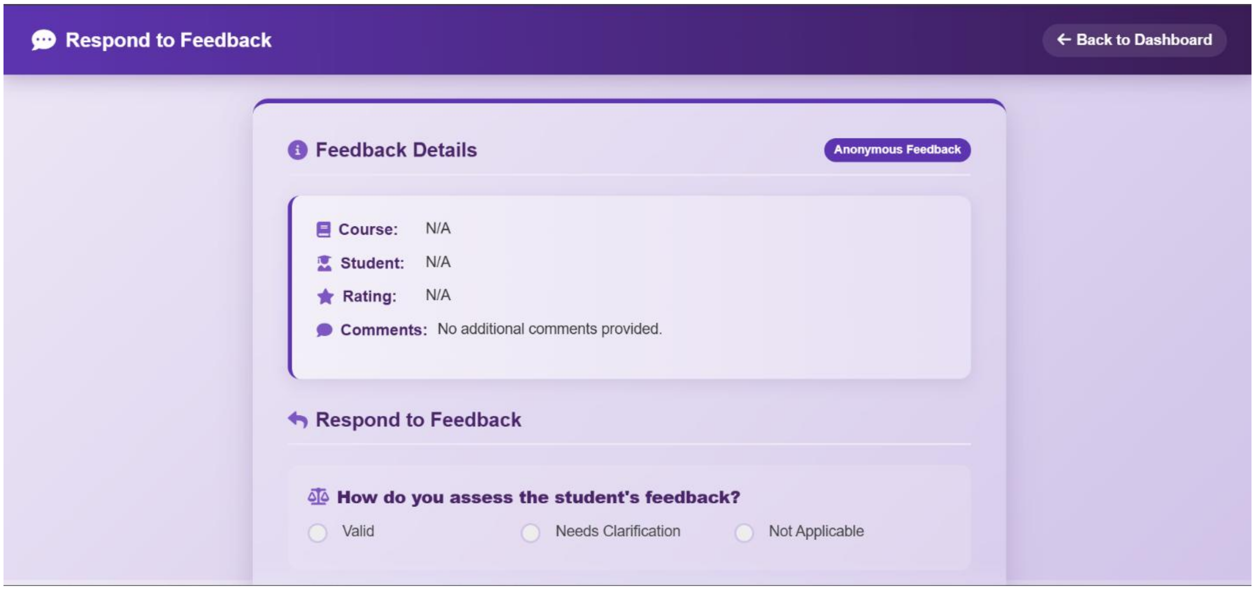
Total Respondents 2

Last Updated 24/04/2025

All Feedback

Unnamed Course N/A N/A
, A
No comment provided
N/A Positive

Unnamed Course N/A N/A
, A
No comment provided
N/A Positive



Feedback Details

Course: N/A

Student: N/A

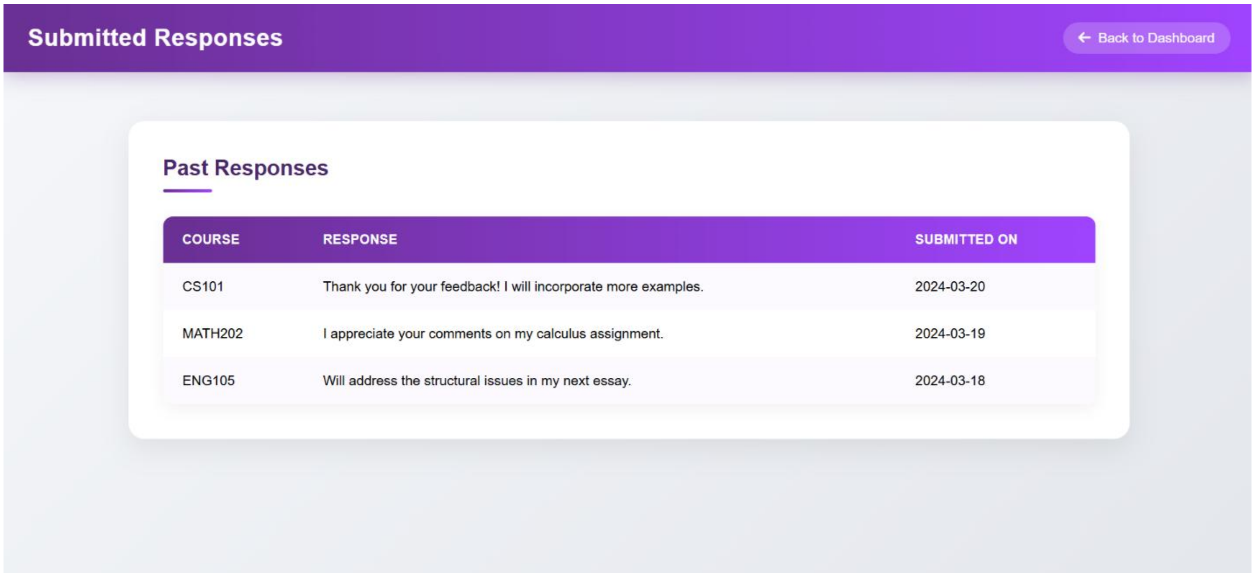
Rating: N/A

Comments: No additional comments provided.

Respond to Feedback

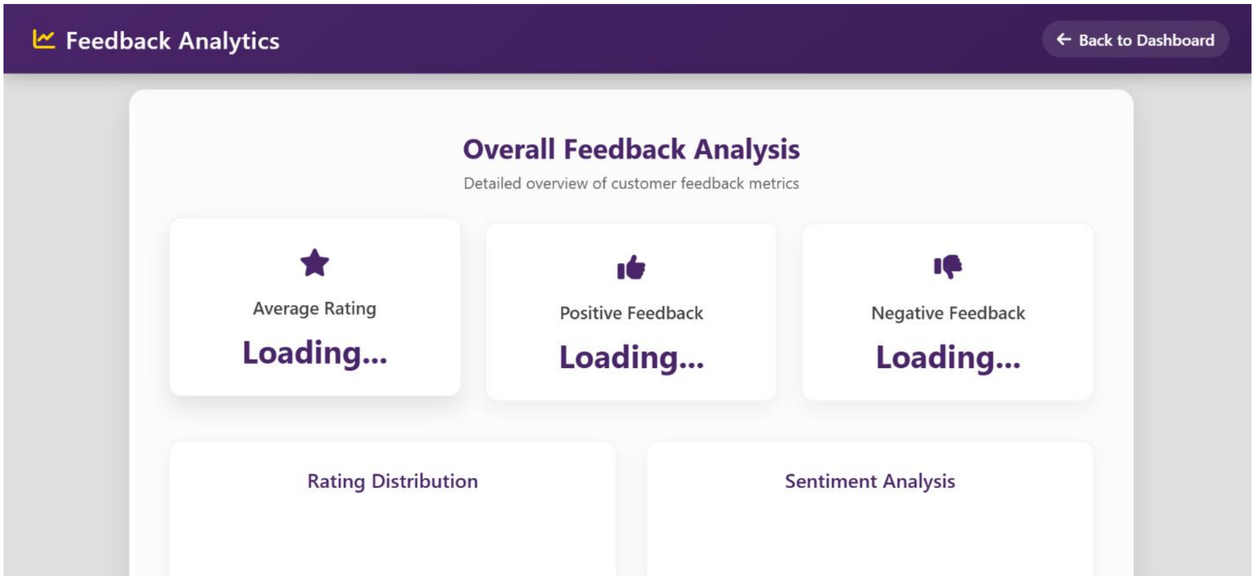
How do you assess the student's feedback?

○ Valid ○ Needs Clarification ○ Not Applicable



Past Responses

COURSE	RESPONSE	SUBMITTED ON
CS101	Thank you for your feedback! I will incorporate more examples.	2024-03-20
MATH202	I appreciate your comments on my calculus assignment.	2024-03-19
ENG105	Will address the structural issues in my next essay.	2024-03-18



Overall Feedback Analysis

Detailed overview of customer feedback metrics

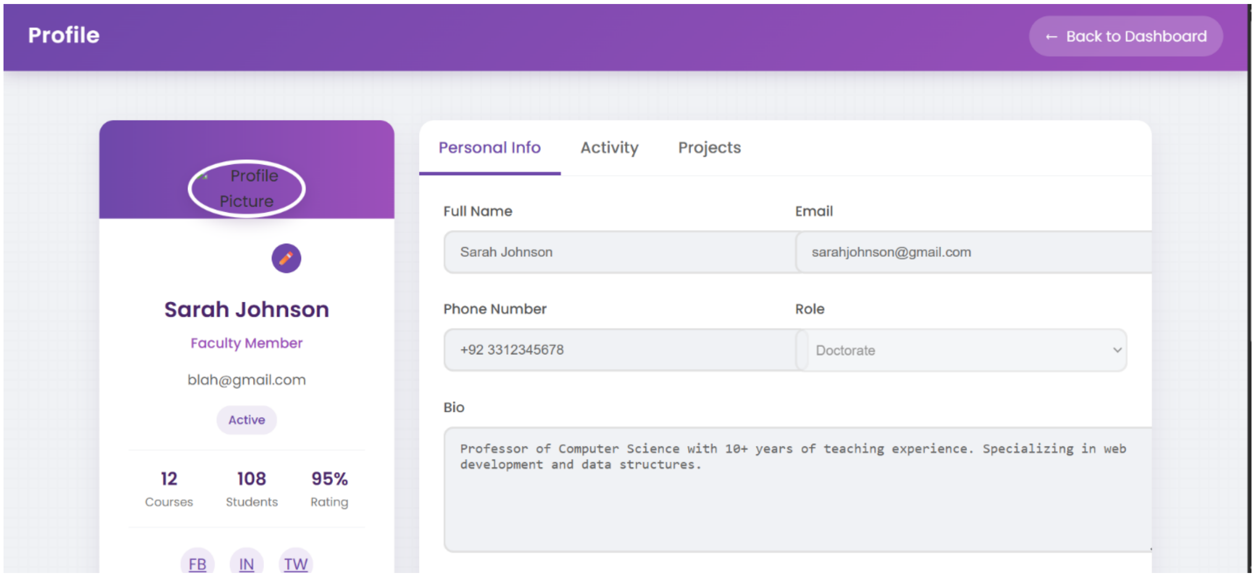
Average Rating
Loading...

Positive Feedback
Loading...

Negative Feedback
Loading...

Rating Distribution

Sentiment Analysis



Sarah Johnson

Faculty Member

blah@gmail.com

Active

12	108	95%
Courses	Students	Rating

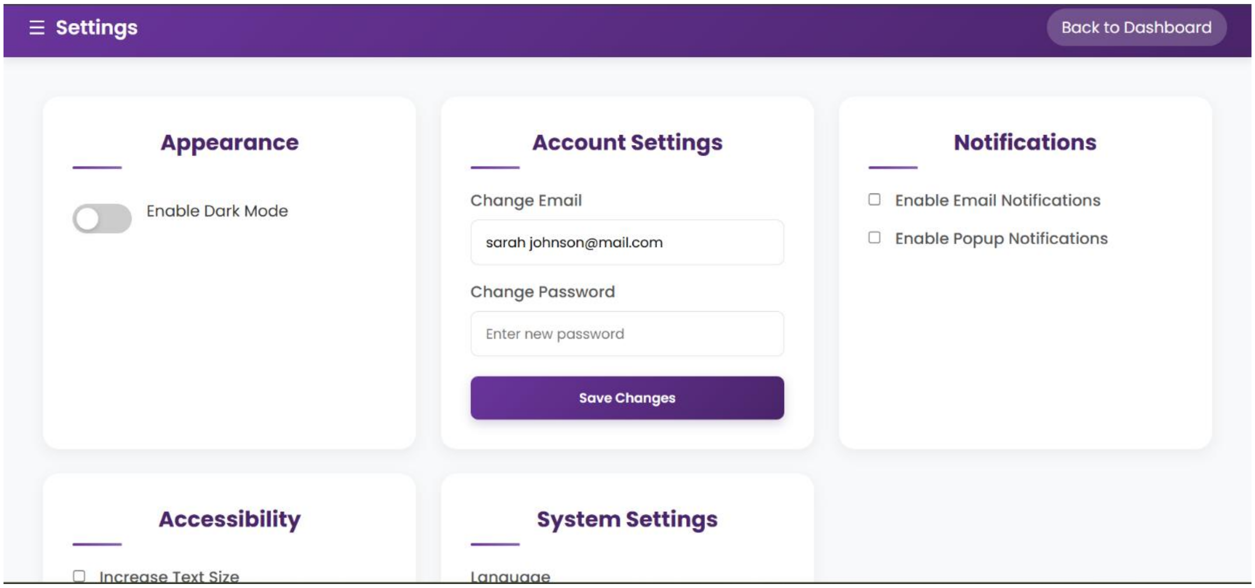
Full Name: Sarah Johnson

Email: sarahjohnson@gmail.com

Phone Number: +92 3312345678

Role: Doctorate

Bio: Professor of Computer Science with 10+ years of teaching experience. Specializing in web development and data structures.



Appearance

Enable Dark Mode

Account Settings

Change Email

sarah johnson@mail.com

Change Password

Enter new password

Notifications

☐ Enable Email Notifications

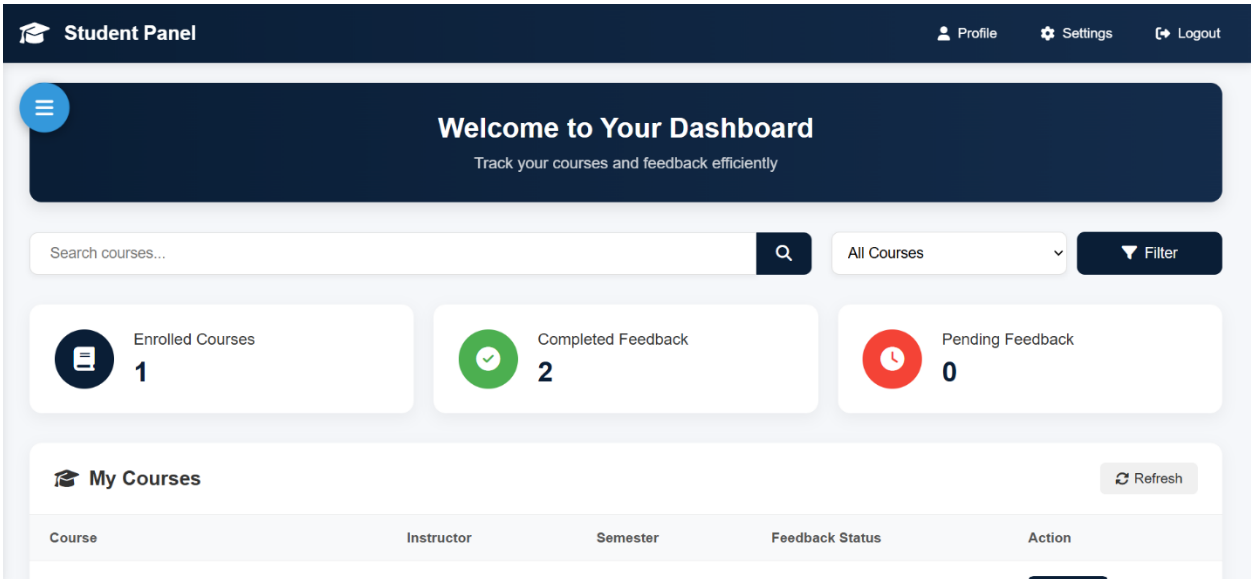
☐ Enable Popup Notifications

Accessibility

☐ Increase Text Size

System Settings

Language



Welcome to Your Dashboard

Track your courses and feedback efficiently

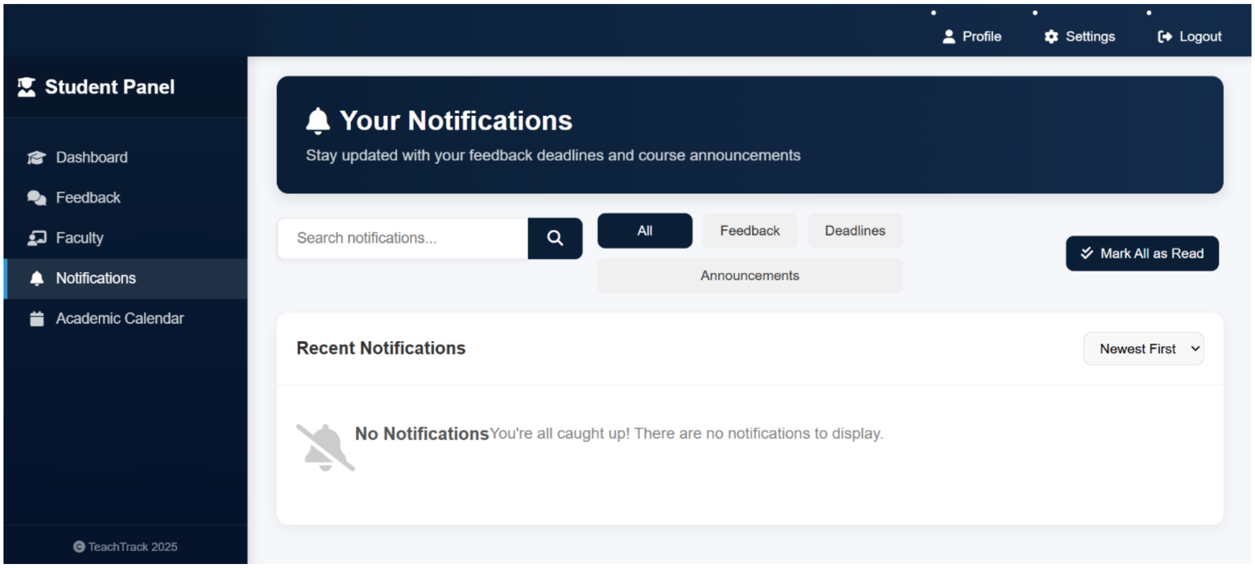
Enrolled Courses 1

Completed Feedback 2

Pending Feedback 0

My Courses

Course	Instructor	Semester	Feedback Status	Action

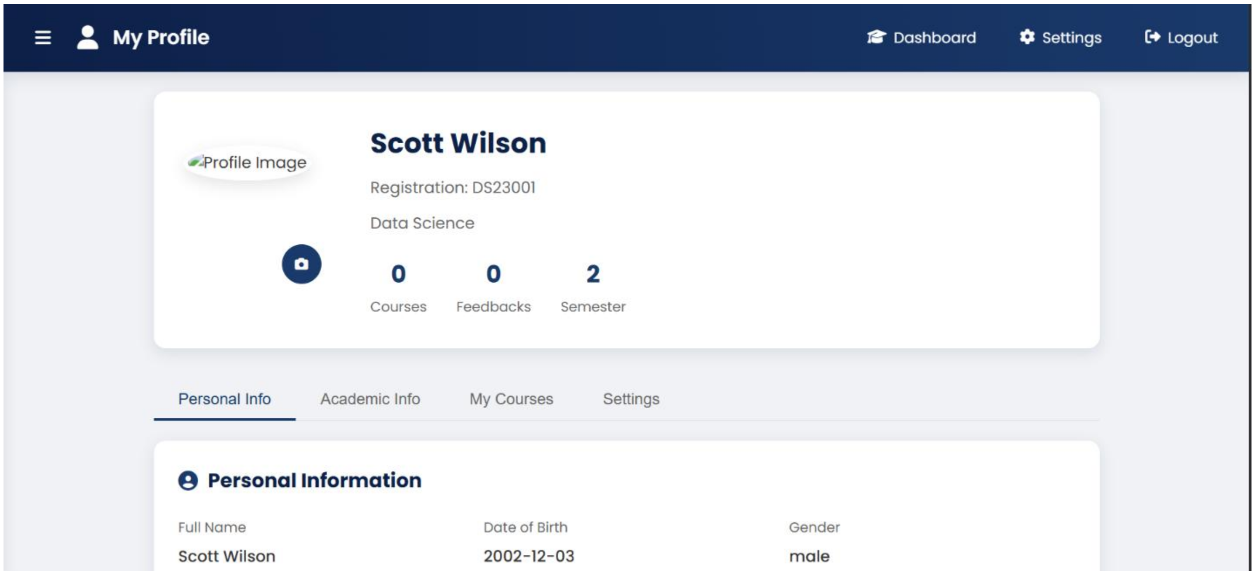


Your Notifications

Stay updated with your feedback deadlines and course announcements

Recent Notifications

No Notifications You're all caught up! There are no notifications to display.



Scott Wilson

Registration: DS23001

Data Science

0 Courses

0 Feedbacks

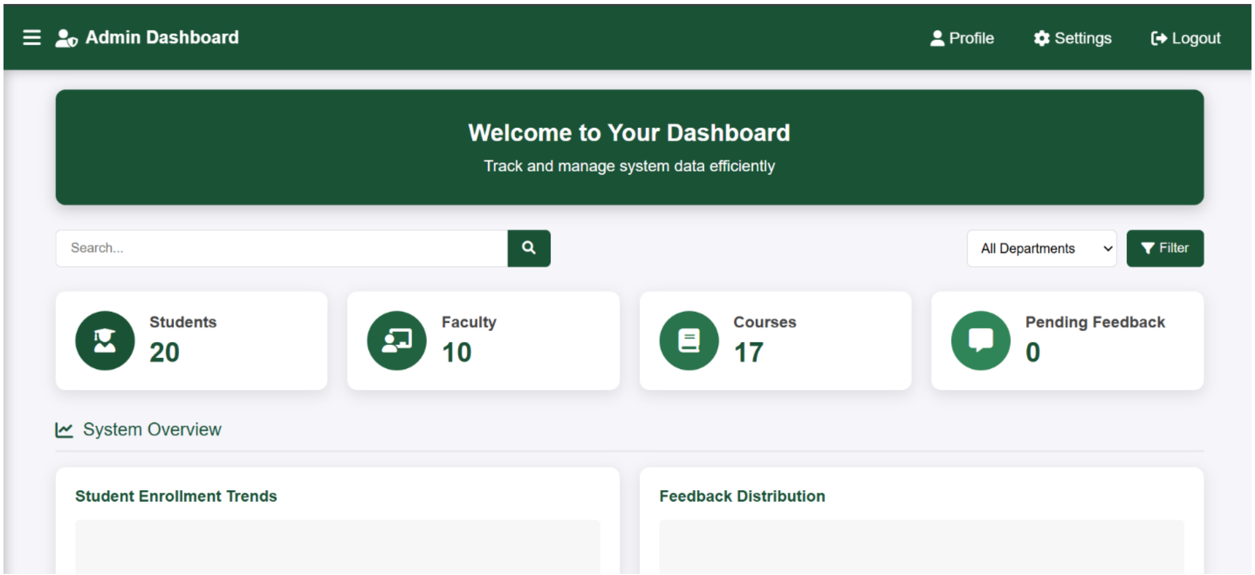
2 Semester

Personal Information

Full Name: Scott Wilson

Date of Birth: 2002-12-03

Gender: male



Welcome to Your Dashboard

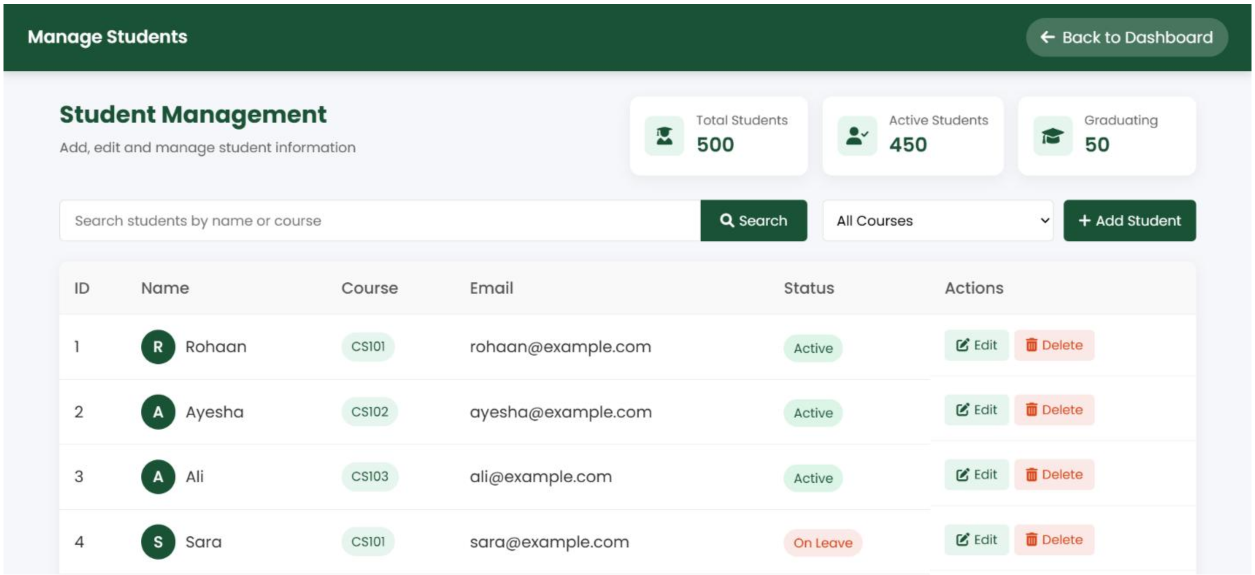
Track and manage system data efficiently

Students	Faculty	Courses	Pending Feedback
20	10	17	0

System Overview

Student Enrollment Trends

Feedback Distribution

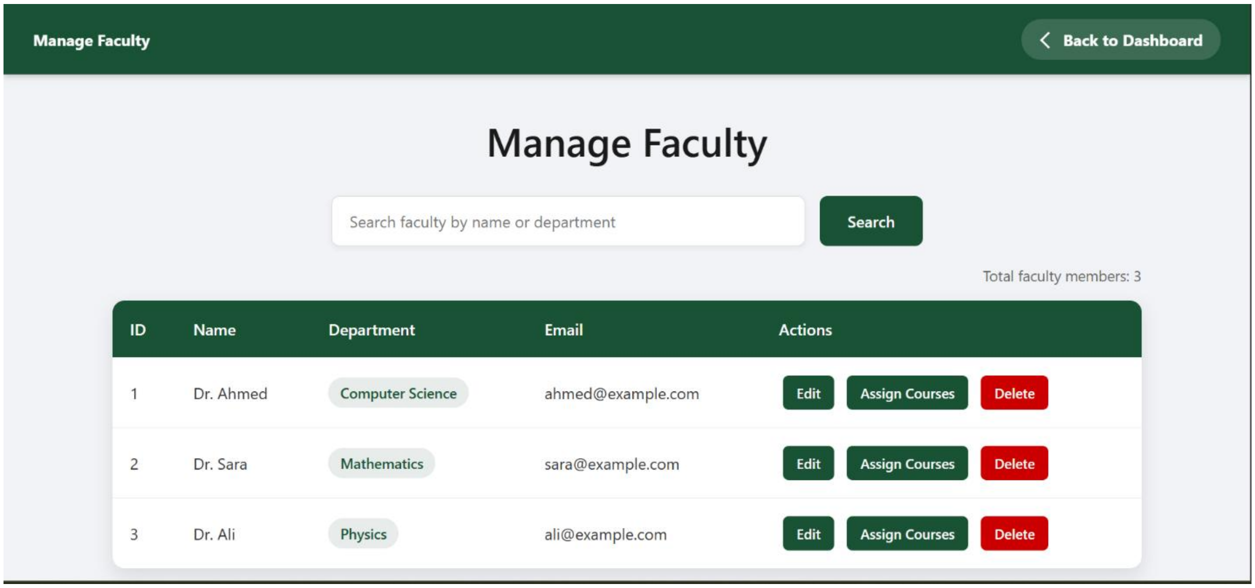


Student Management

Add, edit and manage student information

Total Students	Active Students	Graduating
500	450	50

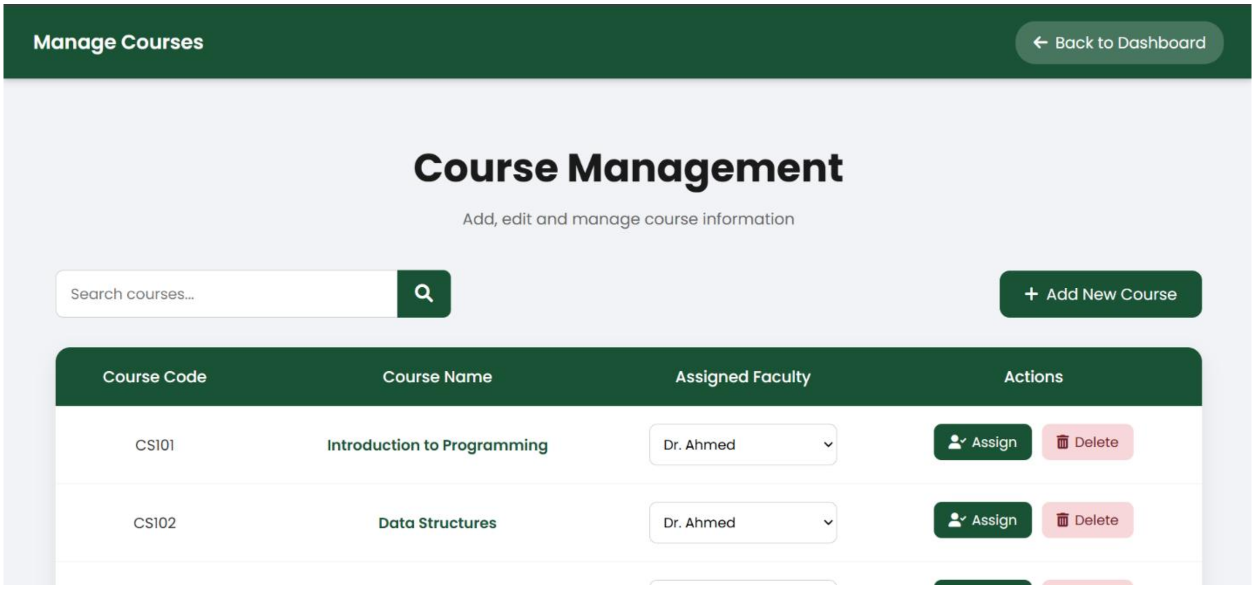
ID	Name	Course	Email	Status	Actions
1	R Rohaan	CS101	rohaan@example.com	Active	Edit Delete
2	A Ayesha	CS102	ayesha@example.com	Active	Edit Delete
3	A Ali	CS103	ali@example.com	Active	Edit Delete
4	S Sara	CS101	sara@example.com	On Leave	Edit Delete



Manage Faculty

Total faculty members: 3

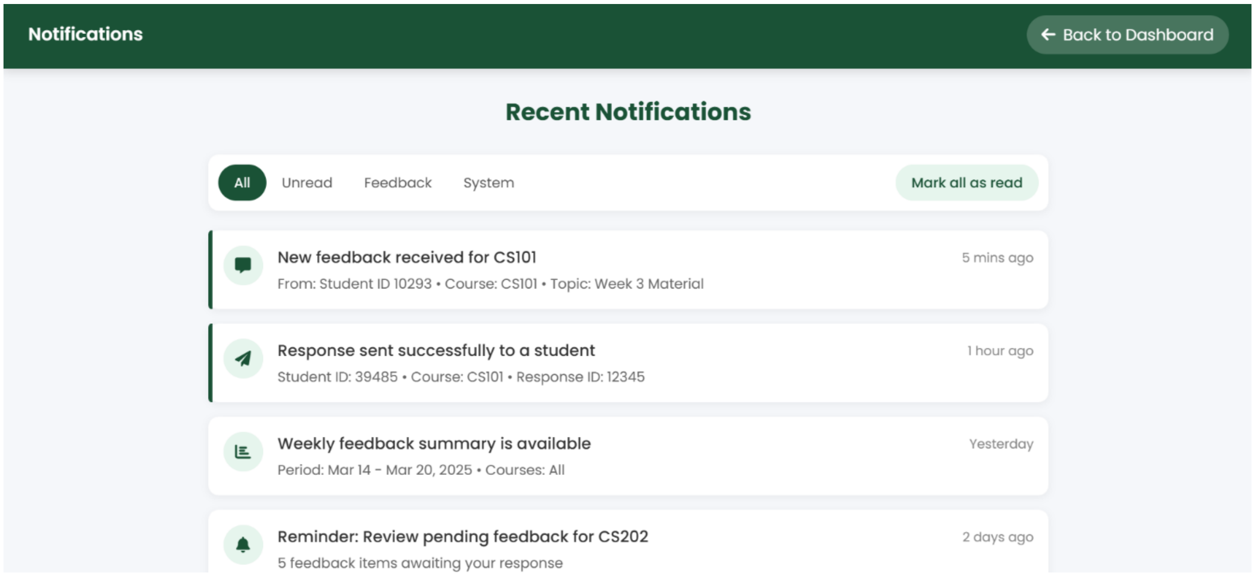
ID	Name	Department	Email	Actions
1	Dr. Ahmed	Computer Science	ahmed@example.com	Edit Assign Courses Delete
2	Dr. Sara	Mathematics	sara@example.com	Edit Assign Courses Delete
3	Dr. Ali	Physics	ali@example.com	Edit Assign Courses Delete



Course Management

Add, edit and manage course information

Course Code	Course Name	Assigned Faculty	Actions
CS101	Introduction to Programming	Dr. Ahmed	Assign Delete
CS102	Data Structures	Dr. Ahmed	Assign Delete



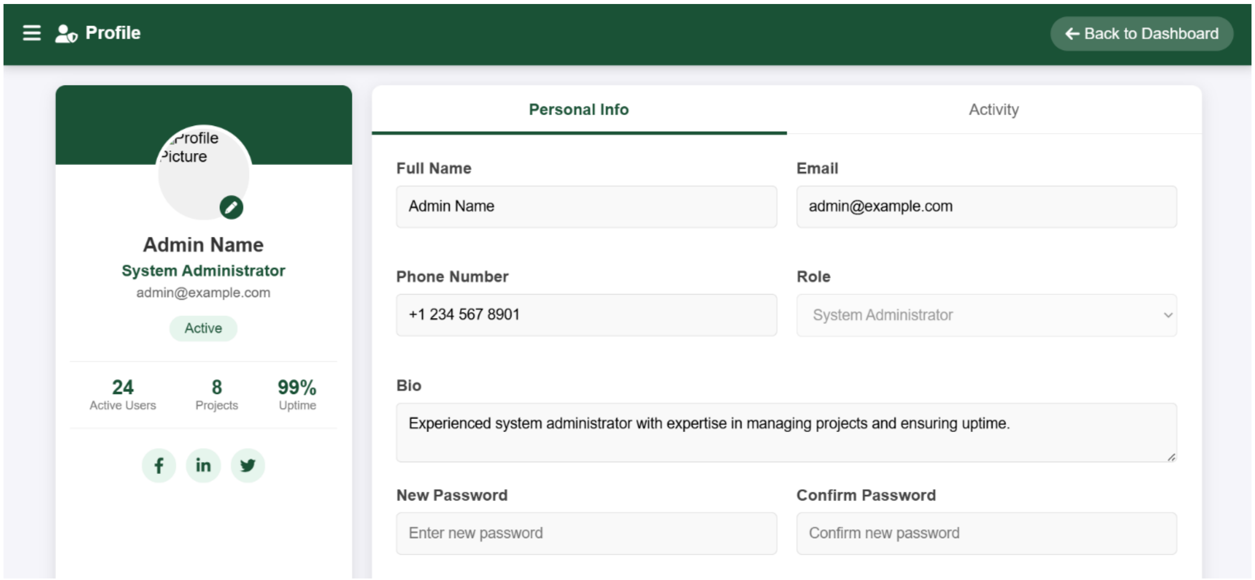
Recent Notifications

New feedback received for CS101 — 5 mins ago
From: Student ID 10293 • Course: CS101 • Topic: Week 3 Material

Response sent successfully to a student — 1 hour ago
Student ID: 39485 • Course: CS101 • Response ID: 12345

Weekly feedback summary is available — Yesterday
Period: Mar 14 - Mar 20, 2025 • Courses: All

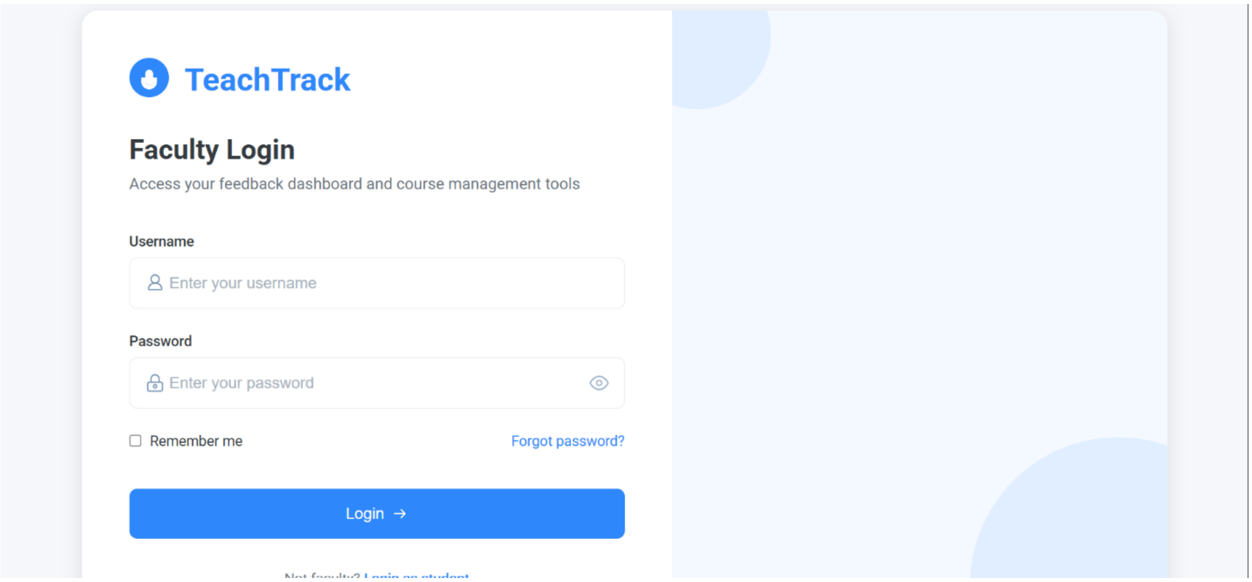
Reminder: Review pending feedback for CS202 — 2 days ago
5 feedback items awaiting your response



Admin Name
System Administrator
admin@example.com
Active

24	8	99%
Active Users	Projects	Uptime

Full Name: Admin Name
Email: admin@example.com
Phone Number: +1 234 567 8901
Role: System Administrator
Bio: Experienced system administrator with expertise in managing projects and ensuring uptime.
New Password: Enter new password
Confirm Password: Confirm new password



TeachTrack

Faculty Login

Access your feedback dashboard and course management tools

Username

Enter your username

Password

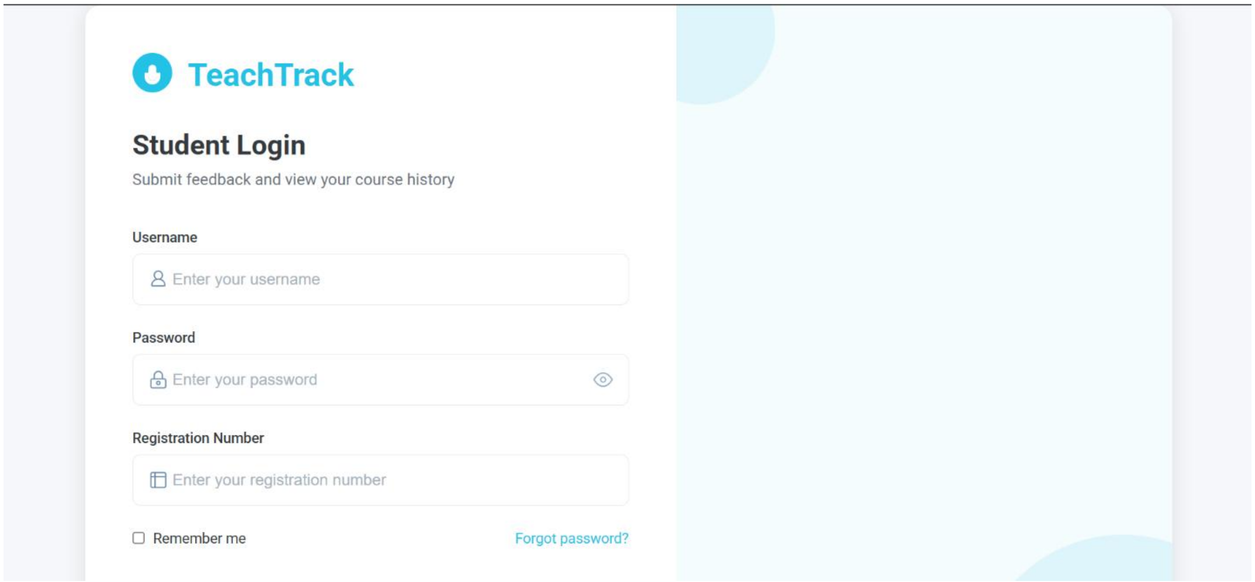
Enter your password

□ Remember me

Forgot password?

Login →

Not faculty? Login as student



TeachTrack

Student Login

Submit feedback and view your course history

Username

Enter your username

Password

Enter your password

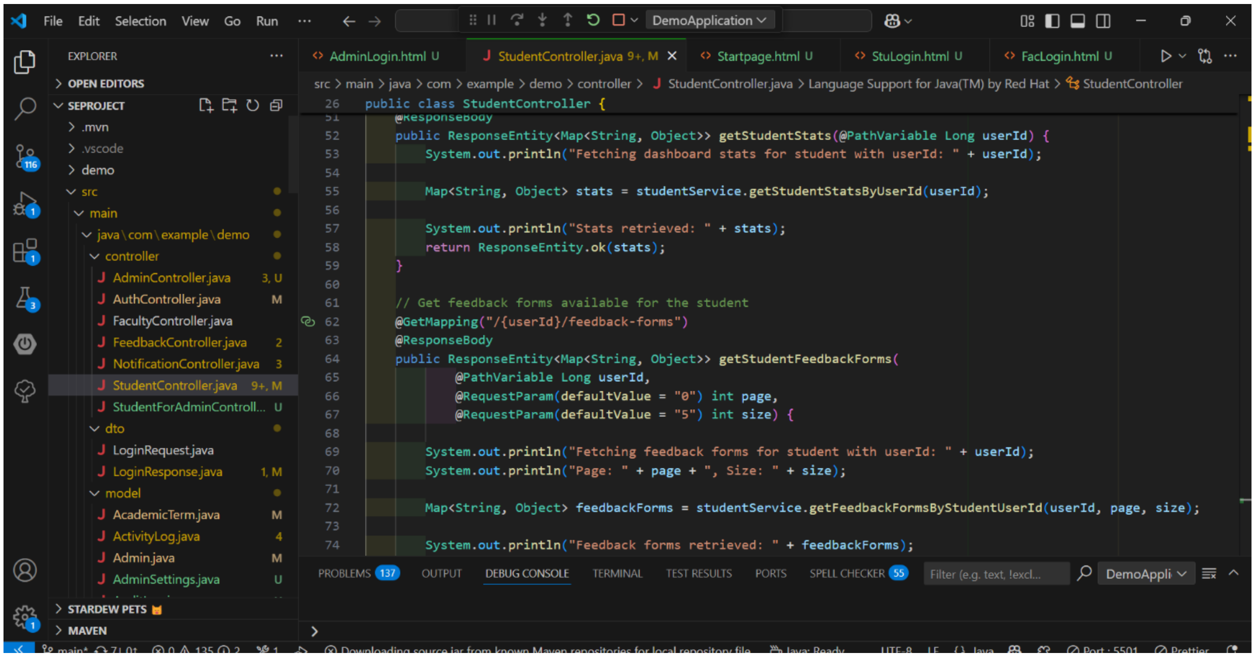
Registration Number

Enter your registration number

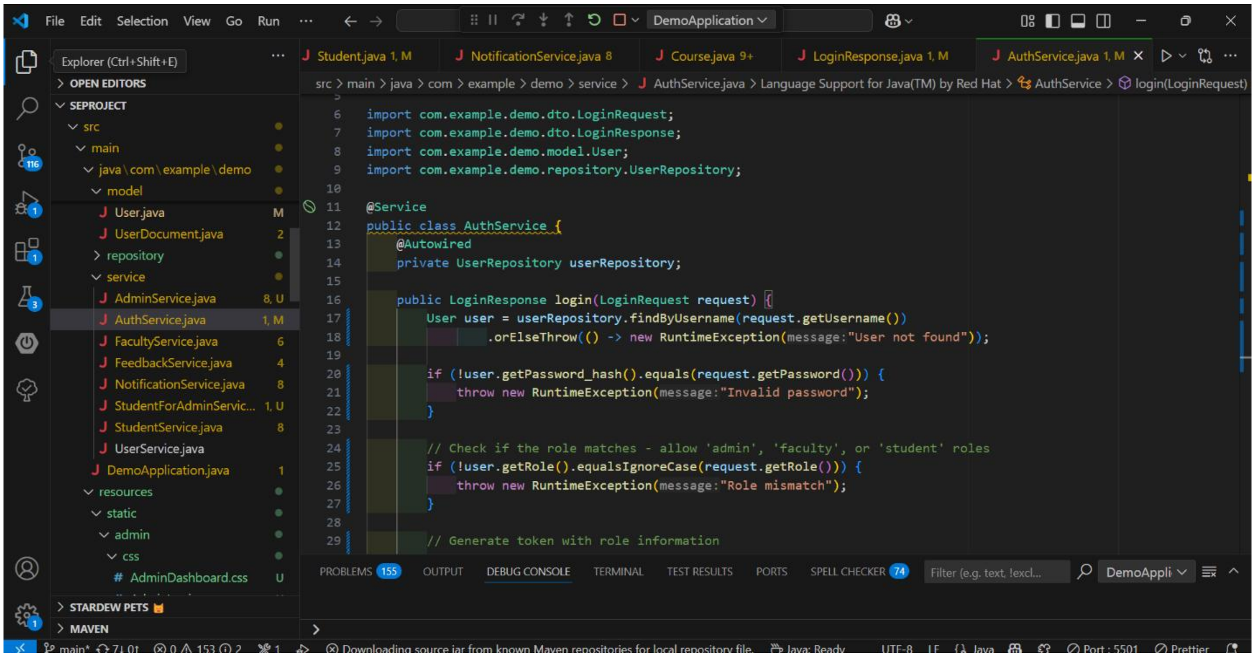
□ Remember me

Forgot password?

ACTUAL IMPLEMENTATION SCREENSHOTS

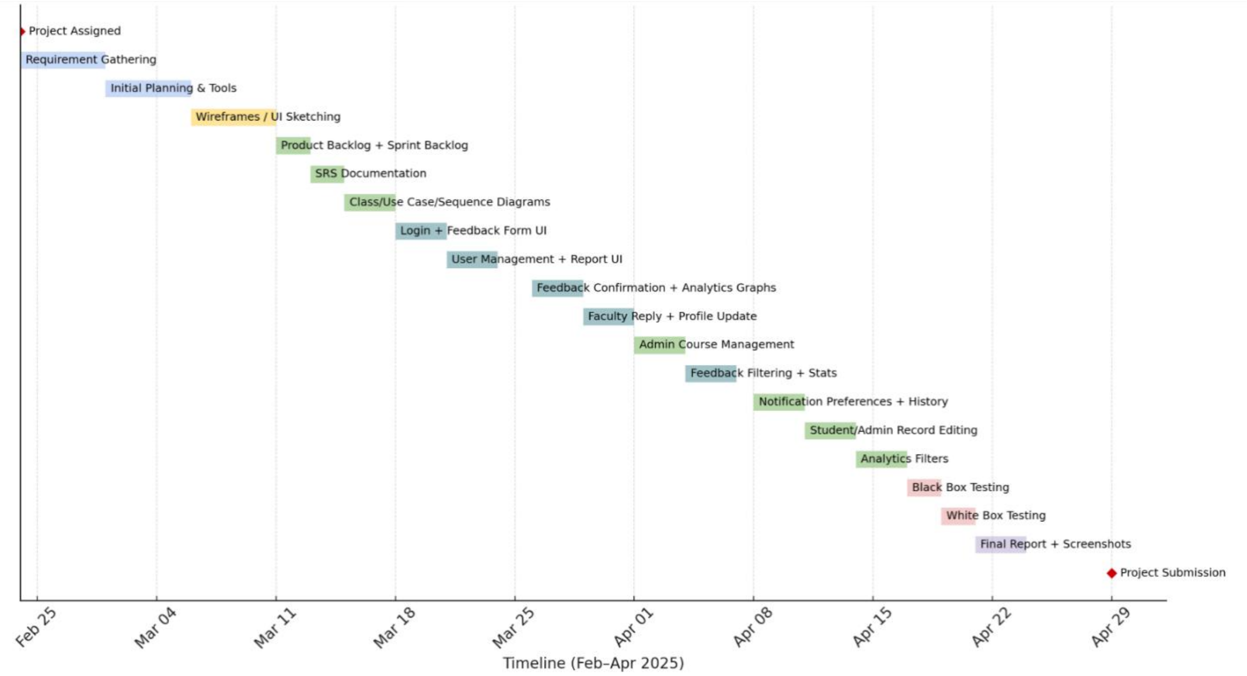


```java
public class StudentController {
    @ResponseBody
    public ResponseEntity<Map<String, Object>> getStudentStats(@PathVariable Long userId) {
        System.out.println("Fetching dashboard stats for student with userId: " + userId);

        Map<String, Object> stats = studentService.getStudentStatsByUserId(userId);

        System.out.println("Stats retrieved: " + stats);
        return ResponseEntity.ok(stats);
    }

    // Get feedback forms available for the student
    @GetMapping("/{userId}/feedback-forms")
    @ResponseBody
    public ResponseEntity<Map<String, Object>> getStudentFeedbackForms(
            @PathVariable Long userId,
            @RequestParam(defaultValue = "0") int page,
            @RequestParam(defaultValue = "5") int size) {

        System.out.println("Fetching feedback forms for student with userId: " + userId);
        System.out.println("Page: " + page + ", Size: " + size);

        Map<String, Object> feedbackForms = studentService.getFeedbackFormsByStudentUserId(userId, page, size);

        System.out.println("Feedback forms retrieved: " + feedbackForms);
```

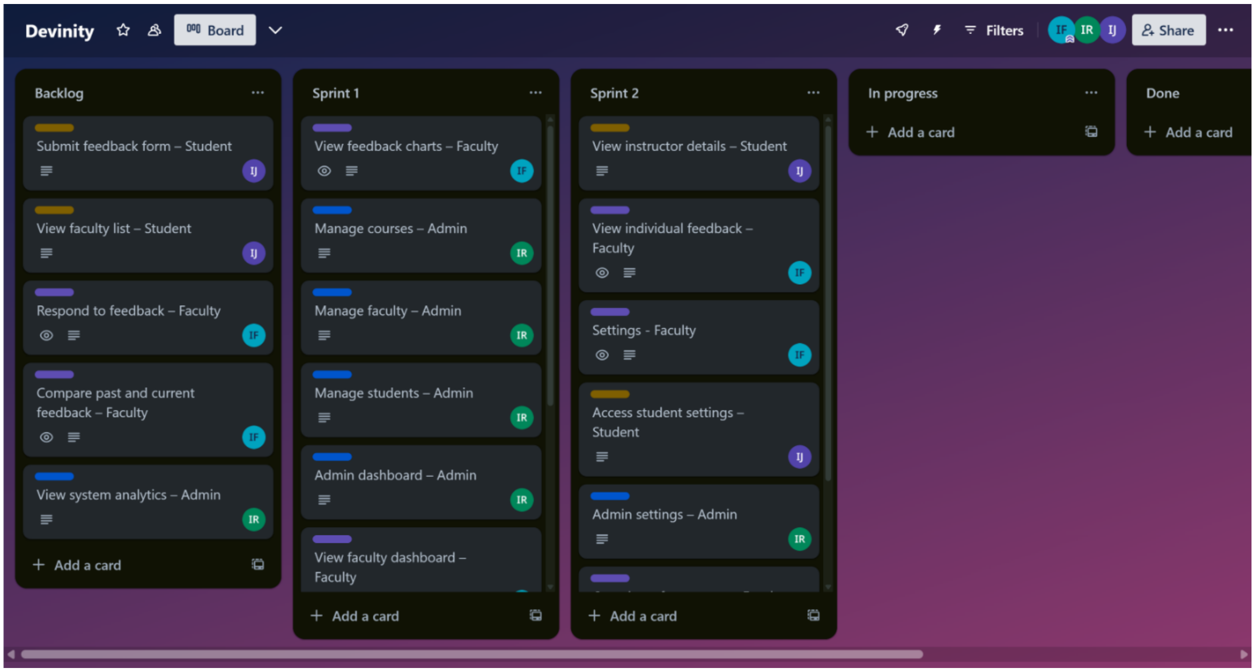


```java
import com.example.demo.dto.LoginRequest;
import com.example.demo.dto.LoginResponse;
import com.example.demo.model.User;
import com.example.demo.repository.UserRepository;

@Service
public class AuthService {
    @Autowired
    private UserRepository userRepository;

    public LoginResponse login(LoginRequest request) {
        User user = userRepository.findByUsername(request.getUsername())
                .orElseThrow(() -> new RuntimeException(message:"User not found"));

        if (!user.getPassword_hash().equals(request.getPassword())) {
            throw new RuntimeException(message:"Invalid password");
        }

        // Check if the role matches - allow 'admin', 'faculty', or 'student' roles
        if (!user.getRole().equalsIgnoreCase(request.getRole())) {
            throw new RuntimeException(message:"Role mismatch");
        }

        // Generate token with role information
```

GANTT CHART FOR THE PROJECT



TRELLO BOARD SCREEN SHOTS



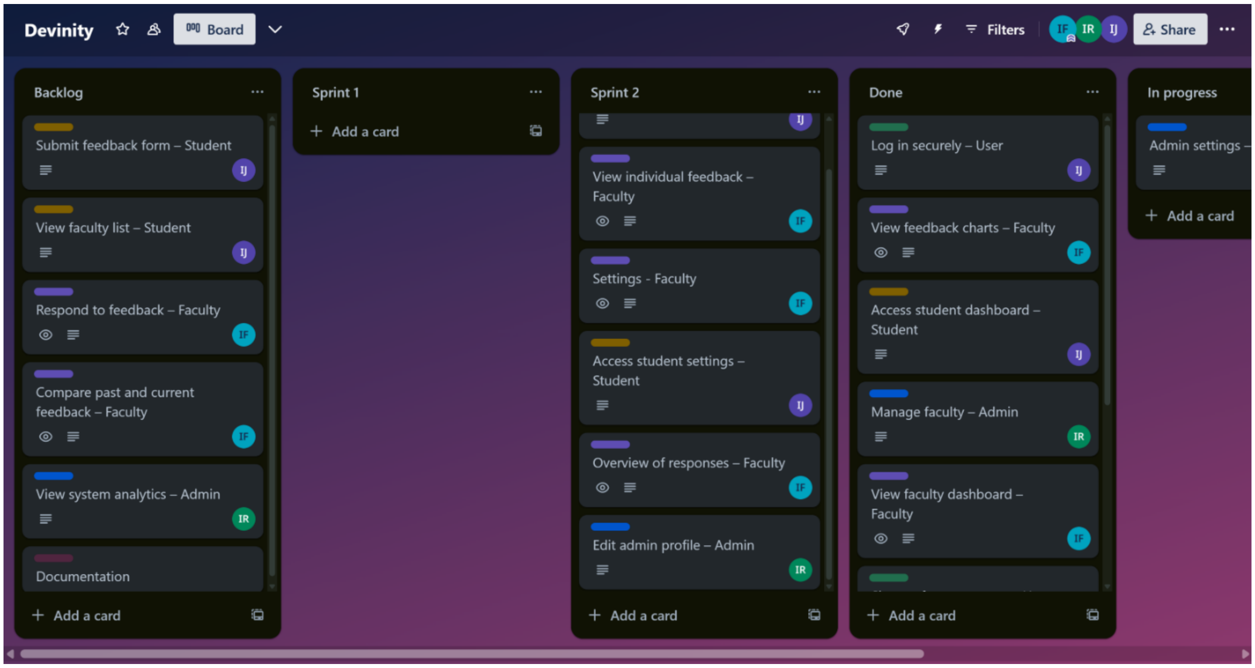
Backlog
- Submit feedback form – Student (IJ)
- View faculty list – Student (IJ)
- Respond to feedback – Faculty (IF)
- Compare past and current feedback – Faculty (IF)
- View system analytics – Admin (IR)
- + Add a card

Sprint 1
- View feedback charts – Faculty (IF)
- Manage courses – Admin (IR)
- Manage faculty – Admin (IR)
- Manage students – Admin (IR)
- Admin dashboard – Admin (IR)
- View faculty dashboard – Faculty
- + Add a card

Sprint 2
- View instructor details – Student (IJ)
- View individual feedback – Faculty (IF)
- Settings - Faculty (IF)
- Access student settings – Student (IJ)
- Admin settings – Admin (IR)
- + Add a card

In progress
- + Add a card

Done
- + Add a card



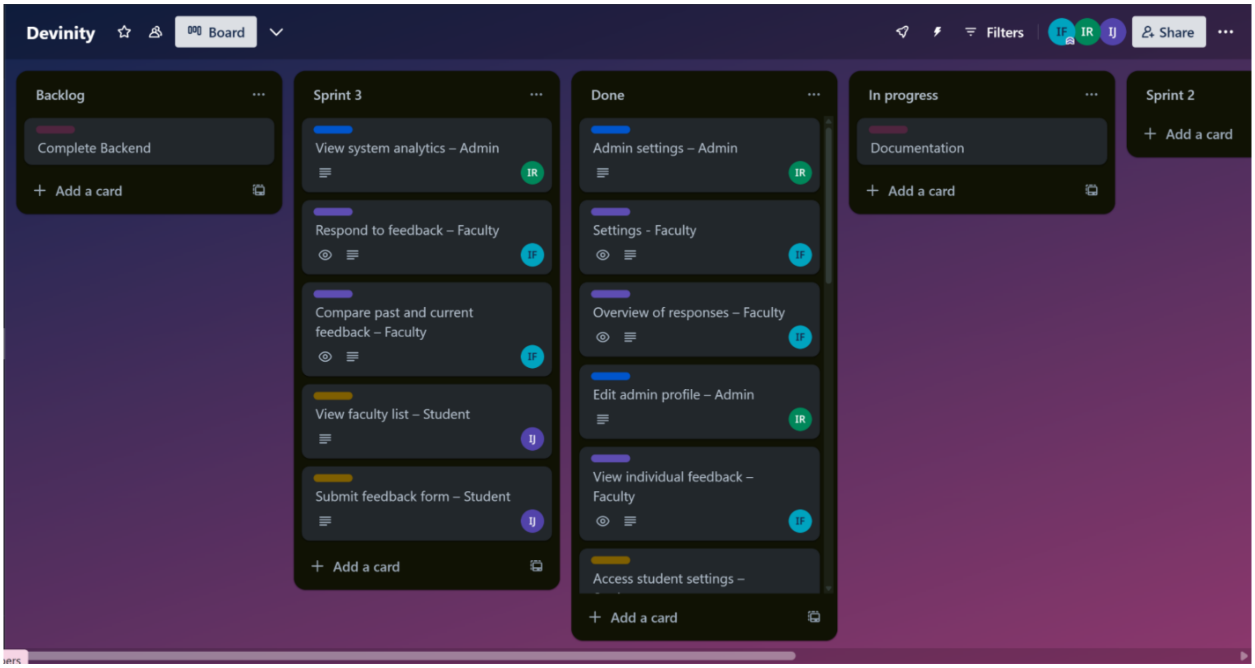
Backlog
- Submit feedback form – Student (IJ)
- View faculty list – Student (IJ)
- Respond to feedback – Faculty (IF)
- Compare past and current feedback – Faculty (IF)
- View system analytics – Admin (IR)
- Documentation
- + Add a card

Sprint 1
- + Add a card

Sprint 2
- View individual feedback – Faculty (IF)
- Settings - Faculty (IF)
- Access student settings – Student (IJ)
- Overview of responses – Faculty (IF)
- Edit admin profile – Admin (IR)
- + Add a card

Done
- Log in securely – User (IJ)
- View feedback charts – Faculty (IF)
- Access student dashboard – Student (IJ)
- Manage faculty – Admin (IR)
- View faculty dashboard – Faculty (IF)
- + Add a card

In progress
- Admin settings –
- + Add a card



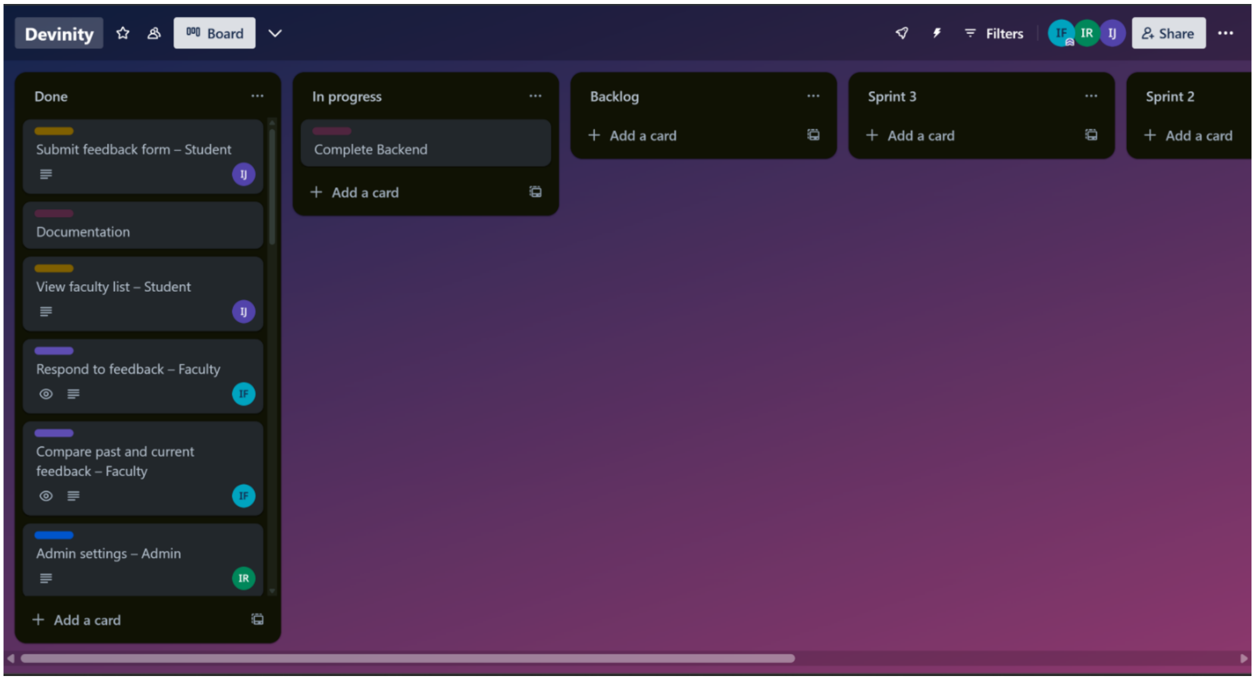
Backlog
- Complete Backend
- Add a card

Sprint 3
- View system analytics – Admin (IR)
- Respond to feedback – Faculty (IF)
- Compare past and current feedback – Faculty (IF)
- View faculty list – Student (IJ)
- Submit feedback form – Student (IJ)
- Add a card

Done
- Admin settings – Admin (IR)
- Settings - Faculty (IF)
- Overview of responses – Faculty (IF)
- Edit admin profile – Admin (IR)
- View individual feedback – Faculty (IF)
- Access student settings –
- Add a card

In progress
- Documentation
- Add a card

Sprint 2
- Add a card



Done
- Submit feedback form – Student (IJ)
- Documentation
- View faculty list – Student (IJ)
- Respond to feedback – Faculty (IF)
- Compare past and current feedback – Faculty (IF)
- Admin settings – Admin (IR)
- Add a card

In progress
- Complete Backend
- Add a card

Backlog
- Add a card

Sprint 3
- Add a card

Sprint 2
- Add a card

TESTCASES - BLACK BOX

SEC01: POST /login – User Login

A. Equivalence Classes

Input Field	**Valid Equivalence Class**	**Invalid Equivalence Class**
username	Non-empty string (minimum 1 character, e.g., "ahmed")	Empty string, special characters (e.g., "$!@"), whitespace, too long (greater than 255 characters)
Password	String of 6–255 characters (e.g., "secret123")	Less than 6 characters, more than 255 characters, empty string

B. Boundary Value Analysis

Input Field	**Just Below Minimum**	**At Minimum**	**Just Above Minimum**	**Nominal Value**	**At Maximum**	**Just Above Maximum**
Username	Empty string → Invalid	1 character → Valid	2 characters → Valid	"ahmed" → Valid	255 characters → Valid	>255 characters → Invalid
password	5 characters → Invalid	6 characters → Valid	7 characters → Valid	"mypassword" → Valid	255 characters → Valid	>255 characters → Invalid

Top 5 Test Cases

TC01 – Valid Login (Nominal Case)

- **Username**: "ahmed"
- **Password**: "secret123"
- **Reason**: Valid input covering the happy path (normal values).
- **Expected Result**: 200 OK, user logged in successfully.

TC02 – Invalid Login (Empty Username)

- **Username**: ""
- **Password**: "secret123"
- **Reason**: Tests invalid **equivalence class** for empty username.
- **Expected Result**: 400 Bad Request, username is required.

TC03 – Invalid Login (Password Below Minimum Length)

- **Username**: "ahmed"
- **Password**: "12345"
- **Reason**: Boundary value analysis — password just below minimum length.
- **Expected Result**: 400 Bad Request, password too short.

TC04 – Valid Login (Username at Maximum Length)

- **Username**: "a"*255
- **Password**: "securepass123"
- **Reason**: Tests boundary condition — username at max allowed length.
- **Expected Result**: 200 OK, login successful.

TC05 – Invalid Login (Username Too Long)

- **Username**: "a"*256
- **Password**: "securepass123"
- **Reason**: Tests invalid input — username exceeds max length.
- **Expected Result**: 400 Bad Request, username too long.

SEC02: GET /userProfile – View User Profile

A. Equivalence Classes

Input Field	Valid Equivalence Class	Invalid Equivalence Class
Authorization Token	Valid token (e.g., valid JWT)	Invalid token (e.g., expired token, malformed token, or missing token)
User Profile Data	Correct data as stored in the database (e.g., username, email, etc.)	Missing or incorrect data (e.g., missing profile field or mismatched values)

B. Boundary Value Analysis

Input Field	Just Below Minimum	At Minimum	Just Above Minimum	Nominal Value	At Maximum	Just Above Maximum

Authorization Token	Missing token (Invalid)	Empty token (Invalid)	Valid token	"Valid JWT Token"	1024 characters (Valid)	>1024 characters (Invalid)

Top 5 Test Cases for /userProfile

TC01 – Valid Request with Proper Authorization and Data

- **Authorization Token**: Valid JWT
- **User Profile Data**: Complete and correct (e.g., username, email)
- **Reason**: Baseline valid case with all correct inputs.
- **Expected Result**: 200 OK, profile data returned successfully.

TC02 – Invalid Request with Missing Token

- **Authorization Token**: Missing
- **User Profile Data**: N/A
- **Reason**: Invalid equivalence class – no authentication provided.
- **Expected Result**: 401 Unauthorized, token required.

TC03 – Invalid Request with Expired Token

- **Authorization Token**: Expired JWT
- **User Profile Data**: N/A
- **Reason**: Invalid equivalence class – token no longer valid.
- **Expected Result**: 401 Unauthorized, token expired.

TC04 – Invalid Request with Token >1024 Characters

- **Authorization Token**: String with 1025 characters
- **User Profile Data**: N/A
- **Reason**: Boundary condition – token exceeds maximum length.
- **Expected Result**: 400 Bad Request, token too long.

TC05 – Invalid Response Due to Incomplete Profile Data

- **Authorization Token**: Valid JWT
- **User Profile Data**: Missing "email" field
- **Reason**: Valid auth but invalid profile data (integrity issue).
- **Expected Result**: 500 Internal Server Error or 422 Unprocessable Entity, profile data invalid.

SEC03: POST /feedback – Submit Feedback

A. Equivalence Classes

Input Field	Valid Equivalence Class	Invalid Equivalence Class
feedback_text	Empty string, only spaces, or special characters (e.g., @#$%)	Empty string, only spaces, or special characters (e.g., @#$%)
rating	Integer between 1 to 5 (representing feedback rating from 1 to 5)	Integer less than 1 or greater than 5 (e.g., 0, 6, -1)
User_id	Valid user ID (e.g., a non-empty string of numbers/characters)	Empty string, invalid format, or non-existing user ID

B. Boundary Value Analysis

Input Field	Just Below Minimum	At Minimum	Just Above Minimum	Nominal Value	At Maximum	Just Above Maximum
feedback_text	Empty string	Empty String	"a" (one character)	Typical feedback (e.g., "Great!")	Long feedback (e.g., 255 characters)	>255 characters (invalid)
rating	0	1	2	3,4	5	> 5 (invalid)
User_id	Empty string	Valid user ID	Valid user ID	Valid user ID	Valid user ID	Invalid user ID

Top 5 Test Cases – POST /feedback

TC01 – Valid Submission (Minimum Input)

- **feedback_text**: ""
- **rating**: 1
- **user_id**: "user123"
- **Reason**: Valid minimum values — lowest rating and empty feedback (allowed).
- **Expected Result**: 201 Created

TC02 – Valid Submission (Maximum Boundaries)

- **feedback_text**: 255-character string

- **rating**: 5
- **user_id**: "user456"
- **Reason**: Tests upper boundary limits for feedback and rating.
- **Expected Result**: 201 Created

TC03 – Invalid Rating (Out of Range – 6)

- **feedback_text**: "App is okay"
- **rating**:
- **user_id**: "user123"
- **Reason**: Rating cant be empty.
- **Expected Result**: 400 Bad Request

TC04 – Invalid Feedback Text (Too Long)

- **feedback_text**: 256-character string
- **rating**: 4
- **user_id**: "user789"
- **Reason**: Feedback exceeds character limit.
- **Expected Result**: 400 Bad Request

TC05 – Invalid User ID (Empty)

- **feedback_text**: "Decent experience"
- **rating**: 3
- **user_id**: ""
- **Reason**: Missing required user ID.
- **Expected Result**: 400 Bad Request

SEC04: GET /feedbacks – View Faculty Feedback

A. Equivalence Classes

Input Field	**Valid Equivalence Class**	**Invalid Equivalence Class**
Facultyid	Valid faculty ID (e.g., a non-empty string of numbers/characters)	Invalid faculty ID (e.g., non-existing faculty ID, empty)

userRole	"admin" (can view all feedback), "student" (can view own feedback)	Invalid role (e.g., "guest", "user")
Filter/sort	Sorting options (e.g., "date", "rating")	Invalid filter or sort parameter (e.g., "price")

B. Boundary Value Analysis

Input Field	Just Below Minimum	At Minimum	Just Above Minimum	Nominal Value	At Maximum	Just Above Maximum
Facultyid	Empty string	Valid faculty ID	Valid faculty ID	Typical valid faculty ID (e.g., "12345")	Very large ID (e.g., 999999999)	Invalid ID (non-existing)
userRole	Invalid role (e.g., "guest")	"student"	"admin"	Typical role (student/admin)	Invalid role (e.g., "user")	Invalid Role
Filter/sort	Invalid filter ("location")	"date"	"rating"	Typical filter (e.g., "date")	Invalid filter (e.g., "alphabet")	Invalid filter

Top 5 Test Cases – GET /feedbacks

TC01 – Admin Requests All Feedback for a Valid Faculty ID

- **facultyId**: "12345"
- **userRole**: "admin"
- **filter/sort**: "date"
- **Reason**: Valid request with authorized role and proper sorting.
- **Expected Result**: 200 OK, all feedbacks sorted by date.

TC02 – Student Views Own Feedback for a Valid Faculty ID

- **facultyId**: "67890"
- **userRole**: "student"
- **filter/sort**: "rating"
- **Reason**: Valid case for student with proper filter.

- **Expected Result**: 200 OK, feedbacks shown only for the logged-in student.

TC03 – Invalid Role Accessing Feedback

- **facultyId**: "12345"
- **userRole**: "guest"
- **filter/sort**: "date"
- **Reason**: Role not authorized to access feedbacks.
- **Expected Result**: 403 Forbidden or 401 Unauthorized

TC04 – Invalid Filter Option

- **facultyId**: "12345"
- **userRole**: "admin"
- **filter/sort**: "location"
- **Reason**: Invalid filter not supported by API.
- **Expected Result**: 400 Bad Request, invalid filter parameter.

TC05 – Invalid Faculty ID (Empty)

- **facultyId**: ""
- **userRole**: "admin"
- **filter/sort**: "date"
- **Reason**: Faculty ID missing — invalid input.
- **Expected Result**: 400 Bad Request, faculty ID required.

SEC05: PUT /updateProfile – Update User Information

A. Equivalence Classes

Input Field	**Valid Equivalence Class**	**Invalid Equivalence Class**
Email	Proper email format with '@' and domain (e.g., "user@example.com")	Malformed email (missing '@', missing domain, e.g., "user@", "example.com")
phoneNumber	10–15 digit numeric string (e.g., "03123456789")	Letters, special chars, too short/too long numbers
Name	Non-empty string	Empty string, only special characters
authToken	Valid token belonging to logged-in user	Expired, invalid, or missing token

B. Boundary Value Analysis

Input Field	Just Below Minimum	At Minimum	Just Above Minimum	Nominal Value	At Maximum	Just Above Maximum
Email	Missing '@' (Invalid)	"a@b.c" → Valid	"a@domain.com" (Valid)	"user@example.com" → Valid	254 characters → Valid	>254 characters → Invalid
phoneNumber	5 digits (Invalid)	10 digits (Invalid)	11 digits (Valid)	"03123456789" (Valid)	15 digits (Valid)	16+ digits (Invalid)
name	Empty (Invalid)	1 char (Valid)	2+ chars (Valid)	“Areeba Riaz” (Valid)	255 chars (Valid)	>255 chars (Invalid)

Top 5 Test Cases – PUT /updateProfile

TC01 – Valid Profile Update with All Correct Fields

- **email**: "user@example.com"
- **phoneNumber**: "03123456789"
- **name**: "Areeba Riaz"
- **authToken**: Valid JWT token
- **Reason**: All fields within valid input ranges.
- **Expected Result**: 200 OK, profile updated successfully.

TC02 – Invalid Email Format (Missing @)

- **email**: "userexample.com"
- **phoneNumber**: "03123456789"
- **name**: "Areeba Riaz"
- **authToken**: Valid JWT token
- **Reason**: Malformed email.
- **Expected Result**: 400 Bad Request, email validation error.

TC03 – Invalid Phone Number (Too Short)

- **email**: "user@example.com"

- **phoneNumber**: "12345"
- **name**: "Areeba Riaz"
- **authToken**: Valid JWT token
- **Reason**: Phone number below minimum digit requirement.
- **Expected Result**: 400 Bad Request, phone number validation error.

TC04 – Missing Authorization Token

- **email**: "user@example.com"
- **phoneNumber**: "03123456789"
- **name**: "Areeba Riaz"
- **authToken**: None
- **Reason**: No token provided.
- **Expected Result**: 401 Unauthorized or 403 Forbidden.

TC05 – Invalid Name (Empty String)

- **email**: "user@example.com"
- **phoneNumber**: "03123456789"
- **name**: ""
- **authToken**: Valid JWT token
- **Reason**: Name is empty, which violates input rules.
- **Expected Result**: 400 Bad Request, name validation error.

SEC06: DELETE /deleteAccount – Delete User Account

A. Equivalence Classes

Input Field	**Valid Equivalence Class**	**Invalid Equivalence Class**
authToken	Valid token associated with an existing, logged-in user	Expired token, malformed token, or token of non-existing user
userId	Matches a user in the database	Non-existent user ID or empty/null

B. Boundary Value Analysis

Input Field	Just Below Minimum	At Minimum	Just Above Minimum	Nominal Value	At Maximum	Just Above Maximum
userId	Empty/null → Invalid	1 char ID → Valid	2 char ID → Valid	Typical UUID → Valid	Max ID length → Valid	>Max length → Invalid

Top 5 Test Cases – DELETE /deleteAccount

TC01 – Successful Deletion with Valid Token and User ID

- **authToken**: Valid JWT for logged-in user
- **userId**: "123e4567-e89b-12d3-a456-426614174000" (valid UUID)
- **Reason**: All fields valid; expected path.
- **Expected Result**: 200 OK, user account deleted successfully.

TC02 – Invalid authToken (Expired Token)

- **authToken**: Expired token
- **userId**: Valid UUID
- **Reason**: Token no longer valid, though userId exists.
- **Expected Result**: 401 Unauthorized, token expired.

TC03 – Non-existent userId

- **authToken**: Valid JWT
- **userId**: "nonexistent-id"
- **Reason**: Token is valid but the user does not exist in DB.
- **Expected Result**: 404 Not Found, user not found.

TC04 – Missing userId

- **authToken**: Valid JWT
- **userId**: "" (empty string)
- **Reason**: Invalid input as user ID is below minimum.
- **Expected Result**: 400 Bad Request, user ID required.

TC05 – Oversized userId (Above Maximum Length)

- **authToken**: Valid JWT
- **userId**: 300-character string
- **Reason**: Violates max length constraint.

- **Expected Result**: 400 Bad Request, user ID too long.

SEC07: GET /courses – List All Courses

A. Equivalence Classes

Input Field	Valid Equivalence Class	Invalid Equivalence Class
userRole	"admin", "faculty", "student" → Authorized users	Unauthorized roles or unauthenticated requests
page	Positive integers (1, 2, 3, …)	Zero, negative values, non-integer inputs
pageSize	Reasonable values like 10, 20, 50	Zero, negative, too large (e.g., 1000+), non-numeric
sortBy	"courseName", "startDate" → Valid sort fields	Invalid fields (e.g., "abc", "xyz")
filterBy	Valid criteria (e.g., "professorId", "semester")	Unsupported filters, malformed query parameters

B. Boundary Value Analysis

Input Field	Just Below Minimum	At Minimum	Just Above Minimum	Nominal Value	At Maximum	Just Above Maximum
page	0 → Invalid	1 → Valid	2 → Valid	5 → Valid	N/A	-1, "a" → Invalid
pageSize	0 → Invalid	1 → Valid	2 → Valid	10, 20, 50 → Valid	100 (assumed max) → Valid	101+ → Invalid (if limited)
sortBy	N/A	Valid field name	N/A	"courseName" → Valid	N/A	"invalidField" → Invalid

Top 5 Test Cases – GET /courses

TC01 – Valid Request by Authorized User

- **userRole**: "student"
- **page**: 1
- **pageSize**: 20
- **sortBy**: "courseName"

- **filterBy**: "semester=Fall2024"
- **Reason**: Fully valid request, all parameters in allowed range.
- **Expected Result**: 200 OK, returns paginated list of courses.

TC02 – Unauthorized Role Accessing Courses

- **userRole**: "guest"
- **page**: 1
- **pageSize**: 10
- **Reason**: Role is not allowed to access course list.
- **Expected Result**: 403 Forbidden, user not authorized.

TC03 – Invalid Page Parameter (Zero)

- **userRole**: "faculty"
- **page**: 0
- **pageSize**: 10
- **Reason**: Page number must be ≥ 1.
- **Expected Result**: 400 Bad Request, invalid pagination.

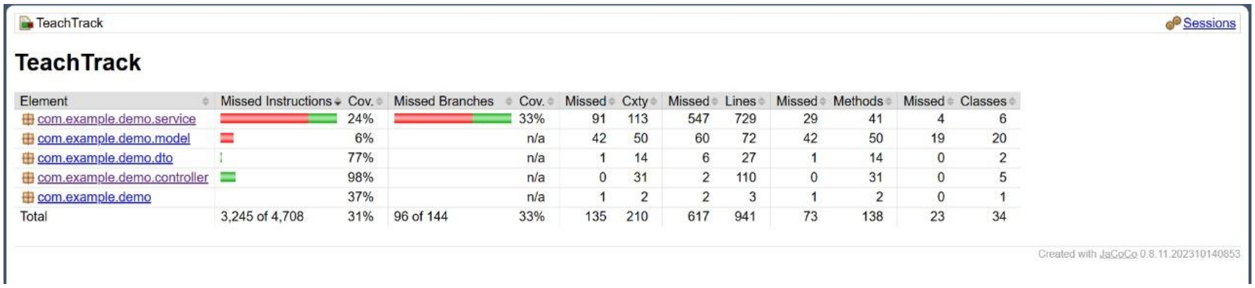
TC04 – Excessive Page Size

- **userRole**: "admin"
- **page**: 1
- **pageSize**: 1001
- **Reason**: Exceeds maximum page size (assumed cap at 100).
- **Expected Result**: 400 Bad Request, page size too large.

TC05 – Invalid sortBy Parameter

- **userRole**: "student"
- **page**: 1
- **pageSize**: 20
- **sortBy**: "xyz"
- **Reason**: xyz is not a valid sort option.
- **Expected Result**: 400 Bad Request, invalid sort field.

TESTCASES - WHITE BOX



TeachTrack

Sessions

TeachTrack

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.example.demo.service		24%		33%	91	113	547	729	29	41	4	6
com.example.demo.model		6%		n/a	42	50	60	72	42	50	19	20
com.example.demo.dto		77%		n/a	1	14	6	27	1	14	0	2
com.example.demo.controller		98%		n/a	0	31	2	110	0	31	0	5
com.example.demo		37%		n/a	1	2	2	3	1	2	0	1
Total	3,245 of 4,708	31%	96 of 144	33%	135	210	617	941	73	138	23	34

Created with JaCoCo 0.8.11.202310140853

WORK DIVISION BETWEEN GROUP MEMBERS

Team Member	Role(S)	Responsibility
Maryam Fasih	Scrum Master – UI Designer	Designed UI Managed Trello Created Diagrams (component, pkg, etc)
Abeer Jawad	Developer – UI/UX Designer	Handled Backend Development Backend Integration White Box & Black Box Testing
Areeba Riaz	Requirement Analyst - Developer	Created Database Schema Black Box Testing Documentation

Although individual roles were assigned, we worked interchangeably, helping each other with frontend, backend, and other tasks whenever needed, ensuring true team collaboration.

LESSON LEARNT BY GROUP

1. Time Management

   We struggled with managing our time properly due to other academic tasks and deadlines. This caused delays in completing some parts of the project. We learned that starting early and making a realistic timeline is very important to avoid last-minute pressure.

2. Role Distribution

   In the beginning, we were not clear about who would do what. This caused confusion and sometimes the same task was done by more than one person. Later, we assigned clear roles to each member, which helped us stay organized and work more smoothly.

3. Communication Issues

Sometimes, we did not update each other on time, which led to misunderstandings. We realized that regular meetings and using shared platforms like Google Drive or WhatsApp for updates helped us stay connected and on track.

4. Teamwork and Flexibility

There were moments when things didn’t go as planned. But by supporting each other and being flexible, we were able to solve problems together. Good teamwork made a big difference in completing our project successfully.